A REDUCED-ORDER MODEL BASED ON PROPER ORTHOGONAL DECOMPOSITION FOR NON-ISOTHERMAL TWO-PHASE FLOWS

A Thesis

by

BRIAN R. RICHARDSON

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2008

Major Subject: Aerospace Engineering

A REDUCED-ORDER MODEL BASED ON PROPER ORTHOGONAL DECOMPOSITION FOR NON-ISOTHERMAL TWO-PHASE FLOWS

A Thesis

by

BRIAN R. RICHARDSON

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Paul Cizmas
Committee Members,	Leland Carlson
	Jean-Luc Guermond
Head of Department,	Helen L. Reed

May 2008

Major Subject: Aerospace Engineering

ABSTRACT

A Reduced-Order Model Based on
Proper Orthogonal Decomposition
for Non-Isothermal Two-Phase Flows. (May 2008)
Brian R. Richardson, B.S.,Texas A&M University
Chair of Advisory Committee: Dr. Paul G. A. Cizmas

This thesis presents a study of reduced-order models based on proper orthogonal decomposition applied to non-isothermal transport phenomena in fluidized beds. A numerical flow solver called Multiphase Flow with Interphase eXchanges (MFIX) was used to generate a database of solution snapshots for proper orthogonal decomposition (POD). Using POD, time independent basis functions were extracted from the data and the governing equations of the numerical solver were projected onto the basis functions to generate reduced-order models. A reduced-order model was constructed that simulates multi-phase isothermal and non-isothermal flow. In the reduced-order models (ROMs) the large number of partial differential equations were replaced by a much smaller number of ordinary differential equations. These reduced-order models were applied to two reference cases, a time extrapolation case and a time-dependent period boundary condition case. Three additional acceleration techniques were developed to further improve computational efficiency of the POD based ROM: 1) Database splitting, 2) Freezing the matrix of the linear system and 3) Time step adjustment. Detailed numerical analysis of both the full-order model, MFIX and the POD-based ROM, including estimating the number of operations and the CPU time per iteration, was performed as part of this study. The results of this investigation show that the reduced-order models are capable of producing qualitatively accurate results with less

than 5% error with a two-order of magnitude reduction of computational costs.

NOMENCLATURE

C_p	–	Constant pressure specific heat
d_{ps}	–	Solid particle diameter
F_{gs}	–	Coefficient for the interphase force between gas and solid phases
g	–	Gravity acceleration
K	–	Diffusivity coefficient
m	–	Number of POD modes
M	–	Number of snapshots
M_w	–	Average molecular weight of gas
N	–	Number of discrete spatial grid points
p	–	Pressure
R	–	Universal gas constant
Re	–	Reynolds number
t	–	Time
T	–	Temperature
(u, v)	–	Components of velocity vector
$\vec{v}$	–	Velocity vector
(x, y)	–	Cartesian coordinates

Greek symbols

α	–	Time-dependent orthogonal coefficients
ϵ_g	–	Void fraction
ϵ_s	–	Solids volume fraction
$\gamma_{g\ell}$	–	Fluids-solids heat transfer coefficient
$\gamma_{R\ell}$	–	Fluids/solids radiative heat transfer coefficient
μ	–	Viscosity
ρ	–	Density
ψ	–	General scalar
$\bar{\bar{\tau}}$	–	Viscous stress tensor
ξ	–	Convection weighting factor

Superscripts

$*$	–	Tentative values
$'$	–	Correction term
o	–	Old values
p	–	Term corresponding to pressure
T	–	Term corresponding to temperature
u	–	Term corresponding to x-direction velocity
v	–	Term corresponding to y-direction velocity

Subscripts

e	–	East face of control volume
E	–	Center of east neighbor cell (i+1,j)
g	–	Gas phase
i	–	ith mode
j	–	jth mode
ℓ	–	Gas or solids phase
n	–	North face of control volume
N	–	Center of north neighbor cell (i,j+1)
nb	–	Neighbor cells
P	–	Center of control volume cell (i,j)
s	–	Solids phase or south face of control volume
S	–	Center of south neighbor cell (i,j-1)
w	–	West face of control volume
W	–	Center of west neighbor cell (i-1,j)
0	–	Zeroth (average) mode

Acronyms

$MFIX$	–	Multiphase Flow with Interphase Exchanges (software)
POD	–	Proper Orthogonal Decomposition
$PODDEC$	–	Proper Orthogonal Decomposition (software)
ROM	–	Reduced-Order Model

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

A. Statement of the Problem

Simulation of two-phase flow problems is important for many applications. Results of two-phase flow simulation is crucial for the design of coal gasifiers, oil refineries and other types of chemical reactors. Reduced-order modeling based on proper orthogonal decomposition (POD) is a computationally efficient alternative to the full-order modeling transport phenomena in a fluidized bed. Full-order modeling for practical problems is time consuming and computationally inefficient. In design applications, when repetitive simulation is needed, full-order modeling may be prohibitively expensive. POD-based reduced-order models (ROMs) achieve better computational efficiency by: (1) reducing the number of equations that need to be solved and (2) replacing the PDEs by ordinary differential equations (ODEs). This research is a continuation of previous work by Tao Yuan who developed POD-based reduced order models to describe the transport phenomena in a fluidized bed. The focus of this research is (1) to develop and implement POD-based ROMs for the calculation of the solids volume fraction correction equation and energy balance equations into the existing ROMs, (2) to develop acceleration techniques for the ROMs, and (3) to investigate the accuracy of the POD models.

1. Transport Phenomena in Fluidized Beds

Fluidization is the phenomenon in which solid particles display fluid-like properties due to the flow of fluids in a solids bed.[16] Figure 1 shows three examples of typical

---

The journal model is *Journal of Propulsion and Power*.

behavior of a fluidized bed. In this study the fluidized bed consisted of a rectangular

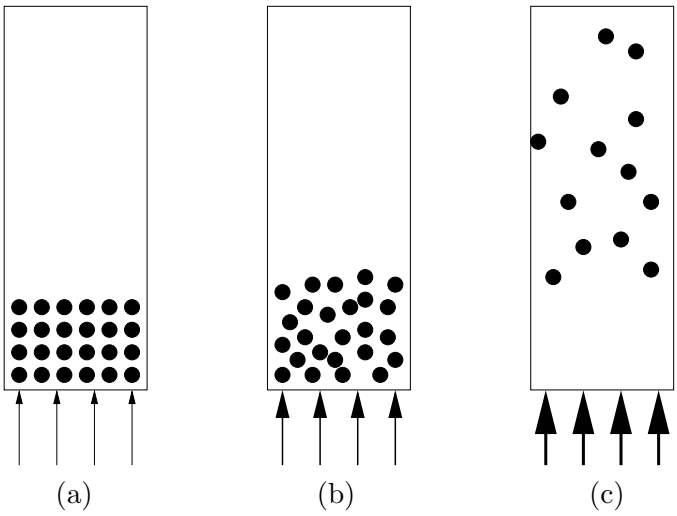


Fig. 1. Typical behavior of a fluidized bed.

control volume containing solid particles and a porous bottom through which gas is injected. The reduced-order model is not limited to this geometry. This particular geometry was suggested by the Department of Energy Lab in Morgantown, WV, the funding agency for this project. At low gas injection velocities as shown in Figure 1(a), the gas percolates through the void spaces between the solid particles and the solids remain completely at rest in a packed bed.[16] When the gas injection velocity increases beyond a certain threshold, called the minimum fluidization velocity, the solid particles begin to display fluid-like properties as shown in Figure 1(b). At this fluidization state, depending on the velocity of the injected gas, the particles could exhibit flow features similar to the gas flow around them or the solids could appear almost at rest at minimum fluidization. If the gas flow velocity is increased beyond the terminal velocity of the solid particles, void spaces between the particles will grow and a gas bubble may form inside the fluidized bed or the particles will be swept out of the bed altogether as shown in Figure 1(c).[16]

For this study, the heat transfer in a fluidized bed are modeled by convection

and diffusion only, without radiation, internal heat sources or chemical reactions. Changes in the phase temperatures in the bed depend on the magnitude of the velocity of the gas through the bed and the diffusivity of the gas and solids phases. Unlike the isothermal case, gas viscosity, gas density and gas pressure vary with time dependent changes in phase temperature. The gas phase is modeled as having the thermochemical properties of air. The solids phase is modeled as having the thermochemical properties of ash, which has a relatively large specific heat value, (C_{ps}) making it a poor conductor of heat. The conduction between the gas phase and solids phase is modeled using a gas-solids heat transfer coefficient, γ_{gs}, corrected for interphase mass transfer.[18] This term is based on the temperature difference between the gas and solids phases and the volume fraction of each phase within a cell. The closing equations for the scalar energy equation can be found in Appendix A.

B. Background

The purpose of this section is to provide a brief summary of background information for this study. This includes an introduction to three techniques used to reduce computational time required to solve systems of equations: 1) Harmonic balance method, 2) Volterra series, and 3) Proper orthogonal decomposition (POD). Of the three methods POD is the most attractive method for computational applications. POD is the only true reduced-order model presented in this section, since POD reduces the size of the system being solved by transforming the governing PDEs into ODEs.

1. Harmonic Balance Method

Harmonic balance method is a way to extend a time linearization approach to nonlinear unsteady problems. The idea of the harmonic balance method is to expand

the flow solution terms in a Fourier series in time. This transforms the solution from the time domain to the frequency domain. Harmonic balance method is not a true reduced-order model because, when applied, the system still retains the same original dimensions of the full-order model.

For an example application of the harmonic balance method consider a differential equation that contains an explicit function of time, τ.

$$\ddot{x} + x = \mu * f(x, \dot{x}, \tau) \tag{1.1}$$

The first approximation of $x(\tau)$ was found using another approximate solution method called the averaging method.[1] The harmonic solution of Equation 1.1 can be written as

$$x(\tau) = a\cos\tau + b\ \sin\tau \tag{1.2}$$

where amplitudes a and b are constants with respect to the periodic solution and are defined in a phase plane that rotates with respect to a representative point. Amplitudes a and b in the rotating phase plane are related to x in the $x-y$ plane by the rotation angle τ and with an angular velocity of unity. The parameter μ is a small number near zero. Equation 1.2 is substituted into the expression of the differential equation given by Equation 1.1. Since a and b are constants over the period, $[0, 2\pi]$ all time derivatives of a and b are set to zero. Equating the coefficients of $sin(\tau)$ and $cos(\tau)$ separately to zero gives the following relationship[1]

$$\int_0^{2\pi} f(a\cos(\tau) + b\ \sin(\tau), a\sin(\tau) + b\ \cos(\tau), \tau)\sin(\tau)d\tau = 0 \tag{1.3}$$

$$\int_0^{2\pi} f(a\cos(\tau) + b\ \sin(\tau), a\sin(\tau) + b\ \cos(\tau), \tau)\cos(\tau)d\tau = 0 \tag{1.4}$$

Higher order approximation is achieved by expanding the periodic solution into a Fourier expansion with unknown coefficients. The Fourier series expansion is again

treated as the assumed solution and substituted into the original governing differential equations. The unknown Fourier coefficients can be fixed by setting the sine and cosine terms of the respective frequencies equal to zero and simultaneously solving the obtained set of equations.[1] The fixed Fourier coefficient terms of the periodic solution represent the frequencies of the assumed solution. Normally only terms containing the harmonic frequency and a few additional frequencies are used to construct an approximate solution. Other additional frequency terms can usually be omitted from the approximation for most cases.[1] Similar to the use of modes in POD, the use of additional frequencies results in a closer approximation but slows numerical computation.

The harmonic balance method has been applied most commonly to low-order problems with non-linear oscillations.[2] A specific application called the method of multiple scales has been used in the simulation of limit cycle oscillation (LCO) for an airfoil in transonic flow.[4] Hall et al. developed a harmonic balance method to model nonlinear aerodynamic problems with large shock motions.[5]

2. Volterra Series Method

The Volterra series method is composed of three major concepts: 1) expressing the system as an infinite sum of homogeneous terms, 2) bounding the time interval to ensure convergence and 3) determining the kernel functions of the Volterra series.

A system that can be described as an infinite sum of homogeneous terms is called a Volterra system.[3] Equation 1.5 represents the response of a time-invariant, non-linear, continuous-time system about an initial state $w(0) = W_0$ due to an input $u(t)$ as a Volterra series.[2]

$$w(t) = h_0 + \int_0^t h_1(t-\tau)u(\tau)d\tau + \int_0^t \int_0^t h_2(t-\tau_1, t-\tau_2)u(\tau_1)u(\tau_2)d\tau_1 d\tau_2 \quad (1.5)$$

$$+\sum_{n=3}^{N}\int_0^t .. \int_0^t h_n(t-\tau_1,...,t-\tau_n)u(\tau_1)...u(\tau_n)d\tau_1...d\tau_n \tag{1.6}$$

Equation 1.5 contains three types of terms. The first term is steady-state and satisfies the initial condition $h_0 = W_0$. The second term in the equation is called the first response term. The term h_1 is the first-order kernel and is identified by measuring the response of the system to a unit impulse at $\tau_1 = 0$. The additional terms contain the second-order kernel, h_2 and nth-order kernels h_n respectively. The second order kernel is identified by measuring the system response in two dimensions resulting from two inputs at two different times.[2] In as far, only continuous-time unit impulse response systems have been presented. However, discrete-time unit impulse responses are less abstract and far more practical for any type of numerical application.

Convergence conditions are needed to ensure that the Volterra system represented by an infinite series is a meaningful one for computation. A Volterra system can only be considered a good approximation if the Volterra system representation converges. Convergence of a Volterra series is dependent on the magnitude of the input function and degree of system non-linearity.[2] Convergence conditions normally bind both the time interval and $u(\tau)$.[3] Usually as the time interval is increased, the input bound, $u(\tau)$, must be made smaller and vise versa.[3] Boyd showed that convergence of the Volterra series is not guaranteed when the maximum value of the input exceeds some system dependent critical value.[8] Finding appropriate bounds for a physical problem is a challenging requirement of the Volterra series method.

Applications of Volterra method reduced-order models have been used primarily to study aeroelastic phenomenon. Identification of the linearized and non-linearized Volterra kernels is a fundamental step in the development of Volterra based ROMs. However, an additional step is needed to transform the kernels into linearized and nonlinear state-space systems.[2] Silva and Raveh successfully used the eigenvalue

realization algorithm (ERA) to generate a linear state-space ROM for aeroelastic applications.[6] Additionally Lucia and Beran used ERA to develop a mixed POD-Volterra ROM algorithm.[7]

3. Reduced-Order Modeling

The purpose of reduced-order modeling is to replace the direct numerical integration of a large number of PDEs by a much smaller number of equations which are usually ODEs. In previous work, researchers have developed many techniques to construct ROMs for a variety of problems in many different fields. The following summary of previous research is focused mainly on proper orthogonal decomposition and flow phenomena investigation.

a. Proper Orthogonal Decomposition

The purpose of this section is to present background information on the mathematical theory of proper orthogonal decomposition. The bulk of this information is heavily based on the work of Cizmas and Palacios and is cited accordingly.

Let us assume we have an ensemble of observations $\{u(x,t_i)\}$, that represent results from experimental observations or numerical analysis. These scalar functions are assumed to form a linear infinite-dimensional Hilbert space L^2 on a spatial domain D.[14] From that ensemble of observations, POD extracts time-independent orthonormal basis functions $\{\phi_k(x)\}$ and time-dependent orthonormal time coefficients $\{\alpha_k(t_i)\}$, such that the reconstruction

$$u(x,t_i) = \sum_k \alpha_k(t_i)\phi_k(x) \tag{1.7}$$

is optimal in the sense that the average least-square truncation error

$$\varepsilon_m = \left\langle \left\| u(x,t_i) - \sum_{j=1}^{m} \alpha_j(t_i)\,\phi_j(x) \right\|^2 \right\rangle \tag{1.8}$$

is a minimum for any given number m of basis functions over all possible sets of basis functions.[14] Herein $||\cdot||$ denotes the L^2-norm given by

$$||f|| = (f,f)^{\frac{1}{2}},$$

where $(,)$ denotes the Euclidean inner product. $\langle\ \cdot\ \rangle$ denotes an ensemble average over a number of observations

$$\langle\ f\ \rangle = \frac{1}{N}\sum_{j=1}^{N} f(x,t_j).$$

The optimum condition specified by Eq. (1.8) is equivalent to finding functions ϕ that maximize the normalized averaged projection of u onto ϕ

$$\max_{\phi\in L^2(D)} \frac{\langle |(u,\phi)|^2 \rangle}{||\phi||^2}, \tag{1.9}$$

where $|\cdot|$ denotes the modulus.[14]

The optimum condition specified by Eq. (1.9) reduces to[10]

$$\int_D \langle u(x)u^*(x')\rangle \phi(x')dx' = \lambda\phi(x). \tag{1.10}$$

The POD basis is therefore composed of the eigenfunctions $\{\phi_j\}$ of the integral equation (1.10). The kernel function of the integral equation (1.10) is the averaged autocorrelation function

$$\langle u(x)u^*(x')\rangle \equiv R(x,x').$$

In practice, the state of a numerical model is only available at discrete spatial grid points. Thus the observations in the ensemble are vectors instead of continuous

functions. The autocorrelation function in the discrete case is replaced by the tensor product matrix[14]

$$R(x,x') = \frac{1}{M}\sum_{i=1}^{M} u(x,t_i)u^T(x',t_i), \tag{1.11}$$

where M is the number of observations contained in the ensemble.

The derivation of the integral equation (1.9) can be generalized to vector-valued functions such as the three-dimensional velocity fields $\mathbf{u}(\mathbf{x},t)$, where $\mathbf{u} = (u,v,w)$ and $\mathbf{x} = (x,y,z)$. In this case, $R(x,x')$ is replaced by

$$\mathbf{R}(\mathbf{x},\mathbf{x}') = \frac{1}{M}\sum_{i=1}^{M} \mathbf{u}(\mathbf{x},t_i)\mathbf{u}^T(\mathbf{x}',t_i). \tag{1.12}$$

The eigenfunctions $\phi_j(\mathbf{x})$ are also vector valued.

C. Summary of Work

This thesis presents the development of a reduced-order model based on proper orthogonal decomposition to model non-isothermal transport phenomena in a fluidized bed. Herein the POD algorithm is applied to the multiphase governing PDEs. This reduced-order model was developed into a numerical algorithm that was used to model a 2-D fluidized bed. The derivation of this numerical algorithm and results from the code application are presented in this thesis.

D. Original Contributions to Work

The reduced-order model based on proper orthogonal decomposition presented in this thesis is a continuation of previous work.[9] The contribution of this author was to apply POD to the solids volume fraction correction equation and energy equation to simulate non-isothermal two-phase flow. The numerical model generated was applied to a fluidized bed simulation to investigate the accuracy and speed-up factor of the

ROM. This investigation included simulating a few physical cases using many different basis function input combinations. Herein, the author chose to present a case that represented an optimal combination of best accuracy and best reduction of computational speed found during this investigation. The results of these investigations are shown in the Results Chapter under the sections Case I and II. Additionally, the author investigated a time extrapolation case and a periodic boundary condition case based on the selected optimal case found in the first investigation, Case 1. The results of these investigations are shown in the Results Chapter under the sections Case III and IV. The author developed three additional acceleration techniques: 1) Database splitting, 2) Freezing the matrix of the linear system and 3) Time step adjustment. These acceleration methods further reduced the computational time required to run the reduced-order model. These techniques were applied to the most computational efficient case found in the first investigation. The results of these acceleration methods are also presented herein in the Acceleration Methods Chapter.

E. Outline of This Thesis

Chapter II describes the governing equations and boundary conditions used to model the transport phenomena in fluidized beds. Chapter III presents the full-order model used to numerically integrate the governing equations. Chapter IV describes the general scheme used to construct POD-based ROMs. Chapter V presents derivation of the POD-based ROMs for approximating the solids volume fraction correction equation and energy balance equation in fluidized beds. Chapter VI presents results generated by applying the ROMs to an isothermal fluidized bed, non-isothermal fluidized bed and time extrapolation cases. The accuracy of these results is discussed and compared with the solution obtained using the full numerical model. Chapter

VII presents acceleration methods developed for the POD-based reduced-order models. The conclusions and future work are presented in Chapter VIII. Appendix A describes the constitutive models used to close the transport equations. Appendix B presents the algorithm for calculating the convection factors in the full numerical model. Appendixes C-E present samples of input files for the POD-based ROMs.

CHAPTER II

PHYSICAL MODEL OF NON-ISOTHERMAL TRANSPORT PHENOMENA IN FLUIDIZED BEDS

This chapter presents the physical model used to describe the physical phenomena in a fluidized bed. First, the basic governing equations of the multi-phase flow are presented. Next, the basic boundary conditions for both the isothermal and non-isothermal models are described.

A. Governing Equations

Below are the mass and momentum equations for non-isothermal transport phenomena in a fluidized bed.

- *Gas mass balance*

$$\frac{\partial \epsilon_g \rho_g}{\partial t} + \bigtriangledown \cdot (\epsilon_g \rho_g \vec{v}_g) = 0 \tag{2.1}$$

- *Solid mass balance*

$$\frac{\partial \epsilon_s \rho_s}{\partial t} + \bigtriangledown \cdot (\epsilon_s \rho_s \vec{v}_s) = 0 \tag{2.2}$$

- *Gas momentum balance*

$$\frac{\partial (\epsilon_g \rho_g \vec{v}_g)}{\partial t} + \bigtriangledown \cdot (\epsilon_g \rho_g \vec{v}_g \vec{v}_g) = -\epsilon_g \bigtriangledown p_g + \bigtriangledown \cdot \bar{\bar{\tau}}_g + \epsilon_g \rho_g \vec{g} + F_{gs}(\vec{v}_s - \vec{v}_g) \tag{2.3}$$

- *Solid momentum balance*

$$\frac{\partial (\epsilon_s \rho_s \vec{v}_s)}{\partial t} + \bigtriangledown \cdot (\epsilon_s \rho_s \vec{v}_s \vec{v}_s) = -\epsilon_s \bigtriangledown p_g - \bigtriangledown p_s + \bigtriangledown \cdot \bar{\bar{\tau}}_s + \epsilon_s \rho_s \vec{g} - F_{gs}(\vec{v}_s - \vec{v}_g) \tag{2.4}$$

In the equations above, ϵ, ρ, and $\vec{v}$ represent the volume fraction, density, and velocity vector respectively. The subscript g represents the gas phase and s, the solids

phase. The gas-phase viscous stress $\bar{\bar{\tau}}_g$, gas-solid drag F_{gs}, granular stress $\bar{\bar{\tau}}_s$, and solid pressure p_s terms found in the momentum equations are needed to close the governing equations. The constitutive models for these variables can be found in Appendix A and are also given in Syamlal *et al.*[17] and Syamlal.[18] The scalar energy balance equations used to model the gas and solids phase temperature fields for a non-isothermal fluidized bed are given by:

- *Gas energy balance*

$$\epsilon_g \rho_g C_{pg} \left(\frac{\partial T_g}{\partial t} + \vec{v}_g \cdot \nabla T_g\right) = -\nabla \cdot \vec{q}_g + \gamma_g (T_s - T_g) - \triangle H_{rg} + \gamma_{Rg} (T_{Rg}^4 - T_g^4) \quad (2.5)$$

- *Solids energy balance*

$$\epsilon_s \rho_s C_{ps} \left(\frac{\partial T_s}{\partial t} + \vec{v}_s \cdot \nabla T_s\right) = -\nabla \cdot \vec{q}_s - \gamma_s (T_s - T_g) - \triangle H_{rs} + \gamma_{Rs} (T_{Rs}^4 - T_s^4) \quad (2.6)$$

where $\vec{q}_\ell$, $\triangle H_{r\ell}$ and $\gamma_{R\ell}$ represent the gas- and solids-phase conductive heat flux, heat of reaction and gas-solids heat transfer coefficient, respectively. The subscript ℓ indicates the gas or solids phase (g or s).

The gas phase is modeled as a gas that obeys the ideal gas law:

$$\rho_g = \frac{p_g \mathcal{M}}{\mathcal{R} T_g} \quad (2.7)$$

where $\mathcal{M}$, $\mathcal{R}$, and T_g denote the average molecular mass of gas, the universal gas constant, and the gas temperature, respectively. The non-isothermal gas phase is assumed to obey the ideal gas law for temperatures present in the studied case. The constant pressure specific heat values C_{pg} and C_{ps} are modeled as those for air and ash particles respectively.

B. Boundary Conditions

Figure 2 below illustrates the basic geometry of a 2-D fluidized bed. The solids particles begin as a packed bed, settled in the lower half of the control volume. Gas is injected at the bottom of the boundary using uniform or non-uniform distributed jet patterns. For the non-isothermal case, the gas entering the inlet is modeled as having a different temperature than the gas in the control volume. The left and right boundaries are modeled as no-slip walls. Constant pressure is specified at the top of the boundary, the gas outlet.

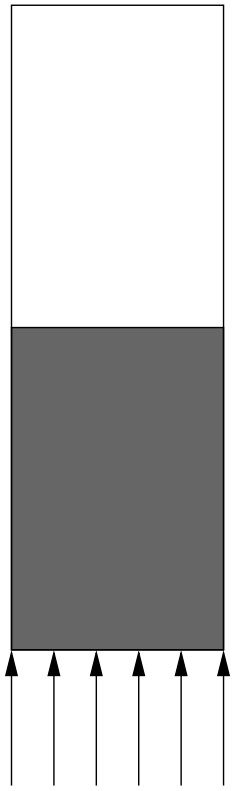


Fig. 2. Basic geometry and boundary conditions.

C. Summary

This chapter presented the governing partial differential equations solved to simulate multi-phase flow. A basic illustration was provided to show the basic boundary conditions used in the investigations presented in this thesis. The next chapter will present the discretized solids volume fraction and energy balance equations. The implementation of these equations into a reduced-order model based on proper orthogonal decomposition is the focus of this thesis.

CHAPTER III

FULL-ORDER MODEL

The full numerical model solves the governing PDEs (2.1)-(2.6) using a fully implicit time marching scheme. The algorithm used to solve the equations was developed at the U.S. Department of Energy's National Energy Technology Laboratory (Syamlal *et al.*[17]). The resulting computer code called MFIX (Multiphase Flow with Interphase eXchanges), is written in FORTRAN 90, is supported on Linux/Unix platforms and can be compiled on 32 or 64 bit processors. It is capable of handling two or three-dimensional Cartesian and cylindrical coordinate systems for uniform or non-uniform grids. It is capable of enforcing free-, partial- or no-slip wall boundary conditions and conducting or non-conducting walls for heat transfer. MFIX numerically solves the gas and solids mass and momentum equations, energy balance and species equations. The output of the code consists of time dependent pressure, gas and solids velocities, temperature, density and composition for a multiphase system. This chapter will present the MFIX grid discretization and the discretized equations relevant to this study.

A. Discretization of the Spatial Domain

MFIX uses a staggered grid arrangement as shown in Figure 3. Scalars are stored at the cell centers. Components of velocity vectors are stored at the cell faces. Equations for scalar variables are solved on the main grid. Equations for velocity components are solved on the staggered grids. If the velocity components and pressure are solved on the same grid, a checkerboard pressure field could result. The staggered grid arrangement is used to prevent this type of unphysical solution.[18] Using the staggered grid arrangement, MFIX uses three grids, which will be described in the following

section, to solve a two-dimensional problem.

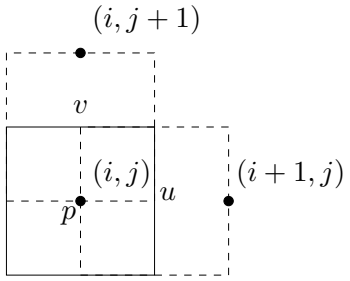


Fig. 3. Grid arrangement in MFIX.

B. Discretized Governing Equations

This section describes the two-dimensional discretized governing equations for the solids volume fraction correction and energy balance in MFIX. The other discretized mass and momentum equations are described in detail in the MFIX Numerical Guide.[18] These equations are not discussed in detail herein because they were not developed into reduced-order models as part of this study. MFIX uses the control volume method to discretize the governing equations.

1. Solids Volume Fraction Correction

Figure 4 shows a control volume for the scalar transport equations used by MFIX. Point P is the center of the control volume. Points E, W, N, and S represent the east, west, north, and south neighbor cells of the control volume. Points e, w, n, and s represent the east, west, north, and south faces of the control volume respectively.

On the computational grid, the scalar values volume fraction (ϵ_ℓ) and density (ρ_ℓ) are stored at the cell centers P, E, W, N, and S. The subscript ℓ indicates

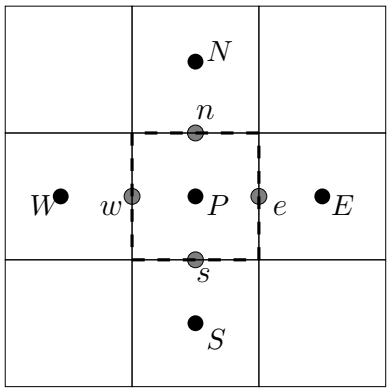


Fig. 4. Control volume for scalar equations.

the phase (g or s). In order to discretize the convection terms, volume fraction and density values at the cell faces e, w, n, and s must be evaluated. MFIX uses a convection weighting factor ξ to calculate the volume fraction and density at each face. Calculation of the convection weighting factor from the down wind factors is presented by Syamlal [18, p. 13-14].

In order to successfully handle dense packing of solids, MFIX uses a solids volume fraction correction equation that includes the effect of solid pressure in the discretized solids mass balance equation.[18] In the algorithm of MFIX, the solids volume fraction correction equation is solved instead of the solid mass balance equation. The solids volume fraction equation is written as[18]

$$a^{\epsilon}_{P}(\epsilon'_s)_P = \sum_{nb} a^{\epsilon}_{nb}(\epsilon'_s)_{nb} + b^{\epsilon}_{P}, \tag{3.1}$$

where the coefficients of a^ϵ and b^ϵ are given by

$$a_E^\epsilon = [(\epsilon_m\rho_m)_e^* e_e (K_s)_E - \xi_e^\epsilon (\rho_s)_E (u_s^*)_e]\, A_e, \tag{3.1a}$$

$$a_W^\epsilon = \left[(\epsilon_m\rho_m)_w^* e_w (K_s)_W + \bar{\xi}_w^\epsilon (\rho_s)_W (u_s^*)_w\right] A_w, \tag{3.1b}$$

$$a_N^\epsilon = [(\epsilon_m\rho_m)_n^* e_n (K_s)_N - \xi_n^\epsilon (\rho_s)_N (v_s^*)_n]\, A_n, \tag{3.1c}$$

$$a_S^\epsilon = \left[(\epsilon_m\rho_m)_s^* e_s (K_s)_S + \bar{\xi}_s^\epsilon (\rho_s)_S (v_s^*)_s\right] A_s, \tag{3.1d}$$

$$\begin{aligned} a_P^\epsilon = {} & (\rho_s)_P[\bar{\xi}_e^\epsilon (u_s^*)_e A_e - \bar{\xi}_w^\epsilon (u_s^*)_w A_w \\ & + \bar{\xi}_n^\epsilon (v_s^*)_n A_n - \bar{\xi}_s^\epsilon (v_s^*)_s A_s] \\ & + (K_s)_P[(\rho_s\epsilon_s^*)_e e_e A_e + (\rho_s\epsilon_s^*)_w e_w A_w \\ & + (\rho_s\epsilon_s^*)_n e_n A_n + (\rho_s\epsilon_s^*)_s e_s A_s] + (\rho_s)_P \tfrac{\Delta V}{\Delta t}, \end{aligned} \tag{3.1e}$$

$$\begin{aligned} b_P^\epsilon = {} & -(\rho_s\epsilon_s^*)_e (u_s^*)_e A_e + (\rho_s\epsilon_s^*)_w (u_s^*)_w A_w \\ & -(\rho_s\epsilon_s^*)_n (v_s^*)_n A_n + (\rho_s\epsilon_s^*)_s (v_s^*)_s A_s \\ & - [(\epsilon_s^*\rho_s)_P - (\epsilon_s\rho_s)_P^o]\, \tfrac{\Delta V}{\Delta t}. \end{aligned} \tag{3.1f}$$

Herein $K_s = \frac{\partial p_s}{\partial \epsilon_s}$. In Eq. 3.1, $\epsilon_s{}'$ is the solids volume fraction correction. The correction $\epsilon_s{}'$, is related to the solids volume fraction by summing the current value of ϵ_s with the correction value [18, p 49]

$$\epsilon_\ell = \epsilon_\ell{}^* + \epsilon_\ell{}'. \tag{3.2}$$

In Eq. 3.4 the subscript ℓ denotes the phase and the superscript $*$ denotes the current or uncorrected value. The volume fraction of the gas and solids phase must always add up to 1. That is to say that ϵ_g and ϵ_s are related by

$$\epsilon_g + \epsilon_s = 1. \tag{3.3}$$

Additionally another parameter called the packed bed void fraction, $\epsilon_g{}^*$, is used in MFIX to calculate granular stress. The packed bed void fraction is the value of the

void fraction in the bed when the solids are at rest and no fluidization is present.

2. Energy Balance Equation

The discretized energy equation is similar to the discretized scalar transport equation [18, p. 54]. Figure 4 shows the control volume for the energy balance equations used by MFIX. On the computational grid, the scalar temperature values, T_ℓ are stored at the cell centers P, E, W, N, and S. The subscript, ℓ, indicates the phase (g or s). As for the solids volume fraction correction equation, MFIX uses a convection weighting factor, ξ, to calculate the temperature at each cell face.

The discretized transport equation for a scalar, ψ, can be written as [18, p. 18, Eq. 3.12]

$$a_P(\psi)_P = \sum_{nb} a_{nb}(\psi)_{nb} + b_P, \tag{3.4}$$

where p and nb denote the cell center and neighbor cell centers respectively. For the energy balance equation $\psi = T$, where T can be the gas or solids phase temperature. The coefficients of the discretized energy balance equation were derived as

$$a_E^{T_\ell} = D_e - \frac{(\xi^{T_\ell})_e}{2}(\epsilon_\ell\rho_\ell)_E(C_{p\ell} + C_{p\ell E})(u_\ell)_e A_e, \tag{3.4a}$$

$$a_W^{T_\ell} = D_w + \frac{(\xi^{T_\ell})_w}{2}(\epsilon_\ell\rho_\ell)_W(C_{p\ell} + C_{p\ell E})(u_\ell)_w A_w, \tag{3.4b}$$

$$a_N^{T_\ell} = D_n - \frac{(\xi^{T_\ell})_n}{2}(\epsilon_\ell\rho_\ell)_N(C_{p\ell} + C_{p\ell N})(v_\ell)_n A_n, \tag{3.4c}$$

$$a_S^{T_\ell} = D_s + \frac{(\xi^{T_\ell})_s}{2}(\epsilon_\ell\rho_\ell)_S(C_{p\ell} + C_{p\ell N})(v_\ell)_s A_s, \tag{3.4d}$$

$$a_p^{T_\ell} = -\Big(\textstyle\sum_{nb}(a^{T_\ell})_{nb} + \frac{(\rho_\ell^o)C_{p\ell}\triangle V}{\Delta t} + \gamma_{Rm}({T_\ell}^o)^3\Big), \tag{3.4e}$$

$$b_p^{T_\ell} = -\Big(\frac{(\rho_\ell^o)C_{p\ell}\triangle V}{\triangle t}({T_\ell}^o) - \triangle H_{Rm}\triangle V + S_{RC\ell}\triangle V\Big), \tag{3.4f}$$

$$(S_{RC\ell})_p = \gamma_{Rm}(T_{Rm}^4 + 3({T_\ell}^o)^4) - 4\gamma_{Rm}({T_\ell}^o)^3 T_\ell, \tag{3.4g}$$

In Equations (3.4a-f), ℓ denotes the phase index (g or s) of the equation. The diffusion terms are modeled using D where [18, p.19, Eq.3.23]

$$D_e = \frac{(K_\ell)_e A_e}{\triangle x_e}. \tag{3.5}$$

The $(K_\ell)_e$ flux term is shown in [18, p. 17, Eq.3.6-3.7]. The subscripts E, W, N and S denote the cell centers of the East, West, North and South neighbor cells, respectively. The subscripts e, w, n and s denote the East, West, North and South face of the cell, respectively. ξ, ϵ, ρ, and C_p are the convection weighting factor, gas void fraction, density and constant volume specific heat, respectively. u_ℓ and v_ℓ are the phase velocities in the x- and y-direction. A is the area of a cell face. The superscript o denotes 'old' variables from the previous time step. The infinitesimal cell volume is represented by $\triangle V$. The radiation source term, $S_{RC\ell}$, is not used in this study.

C. Summary

This chapter presented the full-order model MFIX. The spatial discretization used in MFIX was described. The discretized solids volume fraction correction equation and energy balance equations were presented. In the next chapter these equations will be implemented into a reduced-order model based on proper orthogonal decomposition. For the remainder of this thesis, MFIX will serve as a basis of comparison for the POD based ROM. All error analysis and computational speed-up results for the POD-based ROM is reported in terms of MFIX results.

CHAPTER IV

REDUCED-ORDER MODEL BASED ON PROPER ORTHOGONAL DECOMPOSITION

The purpose of this chapter is to present the devolpment of a reduced-order model based on proper orthoganal decomposition. The presentation of this information is divided into two main sections. The first section presents the general scheme used to generate the POD-based ROMs. The second section presents the application of this scheme to the modeling of isothermal and non-isothermal transport phenomena in a fluidized bed.

A. General Scheme for Reduced-Order Modeling Based on Proper Orthogonal Decomposition

Generation of any reduced-order model based on POD involves three basic steps: (1) generation of a POD database; (2) modal decomposition of the field variables; and (3) Galerkin projection of the basis functions onto the governing equations. To aid discussion in the following sections let us use a general example PDE:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{D}(\mathbf{u}) \tag{4.1}$$

where $\mathbf{u}(\mathbf{x}, t)$ is a state vector; Ω is the spatial domain; $(0, T]$ is the temporal domain and $\mathbf{D}(\mathbf{u})$ is the total or material derivative of u.[9] In this particular example, (4.1) could conceivably represent any of the governing equations (2.1)-(2.6).

1. Database Generation

The database is a collection of data that are solutions of the governing equation or equations. For this example we will assume that the database has been constructed

to represent the solutions of (4.1). The data in the database can be generated by a number of methods including numerical solutions of the governing equations, experimental measurements or a combination of the two. The database can include multiple sets of data in the same temporal domain, but with varying physical properties such as temperature, viscosity and particle size, in order to provide a wider base of data input to expand the reference condition. For equations (2.1)-(2.6), MFIX was used to generate the snapshot database. MFIX was used to compute a number of flow feature snapshots for the pressure, void fraction, velocities and temperature distribution for a single reference condition.

2. Modal Decomposition

The final output of the database generation step above is a set of snapshots represented by $\mathbf{u}(\mathbf{x}, t_i)$, $i \in [1, M]$ where M is the total number of snapshots in the database set. We assume that $\mathbf{u}$ can be decomposed into a time averaged mean $\bar{\mathbf{u}}(\mathbf{x})$ and a time dependent fluctuation $\mathbf{u}'(\mathbf{x}, t)$. The basis functions ϕ_j are the eigenvectors of the matrix $R(\mathbf{x}, \mathbf{x}')$. Using the basis functions, $\mathbf{u}(\mathbf{x}, t)$ is reconstructed as

$$\mathbf{u}(\mathbf{x}, t) = \bar{\mathbf{u}}(\mathbf{x}) + \sum_{j=1}^{M} \alpha_j(t)\phi_j(\mathbf{x}) = \sum_{j=0}^{M} \alpha_j(t)\phi_j(\mathbf{x}), \tag{4.2}$$

where the zeroth basis function $\phi_0(\mathbf{x})$ is the mean $\bar{\mathbf{u}}(\mathbf{x})$ and $\alpha_0(t) \equiv 1$.

Method of snapshots

The database of solutions forms an eigenvalue problem that must be solved to obtained to find the basis functions and time coefficients. A popular technique for finding eigenvalues and eigenvectors of Equation (1.12) is the method of snapshots proposed by Sirovich.[11] When the resolution of the spatial domain N is higher than the total number of snapshots M, the method of snapshots is efficient. The method of

snapshots is based on the fact that the data vectors $\mathbf{u}_i$ and the eigenvectors ϕ_k span the same linear space.[19] Because of this condition, the eigenvectors can be written as a linear combination of the data vectors

$$\phi_k = \sum_{i=1}^{M} v_i^k \mathbf{u}_i, \quad k \in [1, M]. \tag{4.3}$$

If (4.3) is introduced in the eigenvalue problem $\mathbf{R}(\mathbf{x}, \mathbf{x}')\phi(\mathbf{x}) = \lambda\phi(\mathbf{x}')$ we obtain[11]

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v}, \tag{4.4}$$

where $\mathbf{v}^k = (v_1^k, v_2^k, \ldots, v_M^k)$ is the kth eigenvector of (4.4); C is a symmetric $M \times M$ matrix defined by[11]

$$C_{ij} = \frac{1}{M}\left(\mathbf{u}'(\mathbf{x}, t_i),\ \mathbf{u}'(\mathbf{x}, t_j)\right). \tag{4.5}$$

The tensor product matrix R is calculated as

$$R(\mathbf{x}, \mathbf{x}') = \frac{1}{M}\sum_{i=1}^{M} \mathbf{u}'(\mathbf{x}, t_i)\mathbf{u}'^T(\mathbf{x}', t_i).$$

Thus the eigenvectors of the relatively large, $N \times N$, matrix $\mathbf{R}$ are calculated by computing the eigenvectors of the relatively small, $M \times M$, matrix $\mathbf{C}$. In this study, a code called PODDEC developed by Paul Cizmas and Antonio Palacios is used to extract basis functions and time coefficients from a generated database using the method of snapshots.

3. Galerkin Projection

Let us order the eigenvalues such that: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M \geq 0$. Thus the basis functions (eigenvectors, ϕ_j) are also ordered according to their corresponding eigenvalues. If we assume that most of the energy is contained in the first m ($m < M$) POD modes, such that $\sum_{j=1}^{m} \lambda_j \simeq \sum_{j=1}^{M} \lambda_j$, it is reasonable to approximate $\mathbf{u}(\mathbf{x}, t)$

using the first m POD modes:

$$\mathbf{u}(\mathbf{x},t) \simeq \sum_{j=0}^{m} \alpha_j(t)\phi_j(\mathbf{x}). \tag{4.6}$$

Let us next substitute the above approximation of $\mathbf{u}(\mathbf{x},t)$ given by equation 4.6 into the example governing equation (4.1),

$$\sum_{j=1}^{m} \frac{d\alpha_j(t)}{dt}\phi_j(\mathbf{x}) = \mathbf{D}\Big(\sum_{j=0}^{m} \alpha_j(t)\phi_j(\mathbf{x})\Big). \tag{4.7}$$

Projecting (4.7) along the basis function, $\phi_k(\mathbf{x})$ yields

$$\left(\phi_k,\ \sum_{j=1}^{m} \frac{d\alpha_j(t)}{dt}\phi_j(\mathbf{x})\right) = \left(\phi_k,\ \mathbf{D}\Big(\sum_{j=0}^{m} \alpha_j(t)\phi_j(\mathbf{x})\Big)\right). \tag{4.8}$$

Herein transform the governing PDEs into a set of ordinary differential equations,

$$\frac{d\alpha_k}{dt} = F_k(\alpha_1,\ldots,\alpha_m), \quad k \in [1,m], \tag{4.9}$$

where the only unknowns are the time coefficients $\alpha_k(t)$, $k \in [1,m]$. When deriving equation (4.9) from equation (4.8), it is important to note that we have taken advantage of the orthonormal property of the basis functions

$$(\phi_k,\phi_j) = \delta_{kj} = \begin{cases} 1 & \text{if } k = j \\ 0 & \text{if } k \neq j \end{cases},$$

where δ_{kj} is commonly refered to as the Kronecker delta.

The governing equations have been reduced in order by (1) replacing the PDEs (4.1) by a system of ODEs (4.9), and (2) reducing the number of equations from N spatial points to m modes. The linear system of ODEs (4.9) can be integrated using appropriate ODE solvers, *e.g.*, the fourth-order Runge-Kutta method to predict the time history of α_j, $j \in [1,m]$. The state vector $\mathbf{u}(\mathbf{x},t)$ can be reconstructed using the approximation (4.6) with the time coefficients obtained from the ODEs (4.9). The

values of α_j can also be obtained by directly projecting the database onto the jth basis function,

$$\alpha_j^{POD}(t_k) = \left(\phi_j(\mathbf{x}),\ \mathbf{u}(\mathbf{x}, t_k)\right), \quad j \in [1, m],\ k \in [1, M]. \tag{4.10}$$

Since the time coefficients extracted using PODDEC, α_j^{POD}, are obtained by directly projecting the basis functions onto the database, these time coefficients are considered the 'exact' solution of the time coefficients when compared to the time coefficients computed by the reduced-order models. However, this 'exact' property is only true if the output of the ROM and MFIX are compared at the same reference conditions.

B. Reduced-Order Models Based on Proper Orthogonal Decomposition for Transport Phenomena in Fluidized Beds

The purpose of this section is to present the specific derivations and techniques used to generate the POD-based reduced-order models. For ease of explanation the POD-ROM will be refered to by two names: (1) ODEx which simulates isothermal transport phenomena in a fluidized bed and (2) ODEt which models non-isothermal heat transfer and transport phenomena in a fluidized bed. These POD-based reduced-order models are derived from the discretized governing differential equations described in Section A of Chapter II. A summary of the ROMs, governing equations, and unknowns is given in the Table I below.

A POD-based reduced-order model, Hybrid_puv, was developed by Yuan for the x- and y-momentum equations and the discretized gas pressure correction equation.[20] The discretized x- and y-momentum equations and the discretized gas pressure correction equation, were projected onto the basis functions ϕ^{u_ℓ}, ϕ^{v_ℓ}, and ϕ^{p_g}, respectively.

Table I. A summary of governing equations and unknows of POD model.

Model	Governing Equations	Unknowns
ODEx	x-momentum balance	α^{u_g}, α^{v_g}, α^{p}
	y-momentum balance	α^{u_s}, α^{v_s}, α^{ϵ_s}
	pressure correction	
	solids volume fraction	
	correction equation	
ODEt	x-momentum balance	α^{u_g}, α^{v_g}, α^{p}
	y-momentum balance	α^{u_s}, α^{v_s}, α^{ϵ_s}
	pressure correction	α^{T_g}, α^{T_s}
	solids volume fraction	
	correction equation	
	gas and solids energy	
	balance	

Three systems of linear algebraic equations were obtained:[20]

$$\tilde{\mathcal{A}}^{u_\ell}\alpha^{u_\ell} = \tilde{\mathcal{B}}^{u_\ell}, \tag{4.11}$$

$$\tilde{\mathcal{A}}^{v_\ell}\alpha^{v_\ell} = \tilde{\mathcal{B}}^{v_\ell}, \tag{4.12}$$

$$\tilde{\mathcal{A}}^{p_g}\alpha^{p'_g} = \tilde{\mathcal{B}}^{p_g}, \tag{4.13}$$

1. Reduced-order Model Based on Proper Orthogonal Decomposition for the Solids Volume Fraction Correction Equation.

For a two-dimensional, isothermal, multiphase flow, MFIX solves the discretized x- and y-momentum equations for the gas and solids phases, the gas pressure correction equation and the solids volume fraction correction equation. Therefore there are six unknowns per cell in MFIX. These unknown variables are the gas and solids velocity components, u_g, v_g, u_s and v_s, the gas pressure p_g and the void fraction ϵ_g. For each of these field variables a database composed of snapshots has been generated and the POD basis functions $\phi_i^{u_g}$, $\phi_i^{v_g}$, $\phi_i^{u_s}$, $\phi_i^{v_s}$, $\phi_i^{p_g}$ and $\phi_i^{\epsilon_g}$ have been extracted from the data set.

a. ODEx

ODEx is a POD-based ROM generated to model isothermal flow in a fluidized bed. ODEx is derived from the discretized momentum equations, gas pressure correction equation and solids volume fraction correction equation used in MFIX.[18] This section will present the development of the reduced-order model for the solids volume fraction correction equation. The development of the reduced-order models for the x- and y-momentum equations and the gas pressure correction equation was derived by Yuan.[20]

Unlike Hybrid_puv, developed by Yuan,[20] ODEx uses a POD-based algorithm to solve the solids volume fraction correction equation. For clarity, it is important to note that ODEx solves for the time coefficient of the solids volume fraction correction, $\alpha^{\epsilon_s}{}'$. The time coefficient of the correction, $\alpha^{\epsilon_s}{}'$, is summed with the current value of the time coefficient of the solids volume fraction $\alpha^{\epsilon_s *}$ to get the corrected solids volume fraction time coefficient α^{ϵ_s}. The solids volume fraction, α^{ϵ_s}, and void fraction, α^{ϵ_g}, are related by equation 3.3. ODEx uses the basis functions extracted from ϵ_g. This

is possible because of the relationship specified by equation 3.3.

In order for the void fraction to be physically realistic the value should belong to the range $[0, 1]$ for all points in the boundary. Previous work[9] revealed a problem with reconstructing the void fraction using 4.2. Occasionally the void fraction would exceed the value of 1 in some cells of the domain. This was particularly prevalent near the boundary between the solids bed and gas-only freeboard section and at the inlet and outlet boundaries. Modifications made to the solids flux calculations and an increase in the number of modes used to reconstruct the void fraction alleviated the problem described in the previous research. The solids volume fraction correction equation can be written as

$$a_p^{\epsilon}(\epsilon_s^{'})_p = \sum_{nb} a_{nb}^{\epsilon}(\epsilon_s^{'})_{nb} + b_p^{\epsilon}, \tag{4.14}$$

where the a_{nb}^{ϵ},a_p^{ϵ} and b_p^{ϵ} coefficients are defined.[18] $\epsilon_s(x,t)$ is approximated as

$$\epsilon_s(x,t) \cong \phi_0^{\epsilon}(x) + \sum_{i=1}^{m^{\epsilon}} (\alpha_i^{\epsilon^{'}}(t)\phi_i^{\epsilon}(x))_{nb}. \tag{4.15}$$

The solids volume fraction is the sum of the tentative value of the solids volume fraction, ϵ_s^*, and the correction value $\epsilon_s^{'}$. It was assumed that the first mode of the basis functions ϕ_0^{ϵ}, the mean value of ϵ_s, is equal to the tentative value of the solids volume fraction ϵ_s^*. Therefore ϵ_s can be written in terms of the basis functions and time coefficients.

$$\epsilon_s(x,t) \cong \epsilon_s^* + \epsilon_s^{'} = \phi_0^{\epsilon}(x) + \sum_{i=1}^{m^{\epsilon}} (\alpha_i^{\epsilon^{'}}(t)\phi_i^{\epsilon}(x))_{nb}. \tag{4.16}$$

Dropping the x and t notation, the solids volume fraction correction in terms of the basis functions and time coefficients can be written as

$$\epsilon_s^{'} \cong \sum_{i=1}^{m^{\epsilon}} \alpha_i^{\epsilon^{'}} \phi_i^{\epsilon}. \tag{4.17}$$

Substituting (4.17) into (4.14) and projecting the new equation on the basis function, ϕ_k^ϵ, yields the reduced-order solids volume fraction correction equation:

$$a_p^\epsilon\Big(\sum_{i=1}^{m^\epsilon}\alpha_i^{\epsilon^{'}}\phi_i^\epsilon,\phi_k^\epsilon\Big)_p=\sum_{nb}a_{nb}^\epsilon\Big(\sum_{i=1}^{m^\epsilon}\alpha_i^{\epsilon^{'}}\phi_i^\epsilon,\phi_k^\epsilon\Big)_{nb}+\Big(b_p^\epsilon,\phi_k^\epsilon\Big). \tag{4.18}$$

Using (3.1)(a-f) to expand the coefficients a_{nb}, a_p and b_p of (4.18) yields

$$\begin{aligned}
&(\rho_\ell)_P\Delta V\cdot\sum_{i=1}^{m^{\epsilon_\ell}}\dot{\alpha}_i^{\epsilon_\ell}(\phi_i^{\epsilon_\ell},\phi_k^{\epsilon_\ell})=\\
&-\sum_{i=0}^{m^{\epsilon_\ell}}\sum_{j=0}^{m^{u_\ell}}\Big((\rho_\ell\epsilon_\ell)_e e_e[(K_m)_P\phi_i^{\epsilon_\ell}-(K_m)_E\phi_{i,e}^{\epsilon_\ell}]\\
&+[\xi_e^{\epsilon_\ell}(\rho_\ell)_E\phi_{i,e}^{\epsilon_\ell}+\bar{\xi}_e^{\epsilon_\ell}(\rho_\ell)_P\phi_i^{\epsilon_\ell}]\phi_{j,e}^{u_\ell}\alpha_j^{u_\ell}\Big)\Delta y+\\
&+\sum_{i=0}^{m^{\epsilon_\ell}}\sum_{j=0}^{m^{u_\ell}}\Big((\rho_\ell\epsilon_\ell)_w e_w[(K_m)_P\phi_i^{\epsilon_\ell}-(K_m)_W\phi_{i,w}^{\epsilon_\ell}]\\
&+[\bar{\xi}_w^{\epsilon_\ell}(\rho_\ell)_W\phi_{i,w}^{\epsilon_\ell}+\xi_w^{\epsilon_\ell}(\rho_\ell)_P\phi_i^{\epsilon_\ell}]\phi_{j,w}^{u_\ell}\alpha_j^{u_\ell}\Big)\Delta x-\\
&-\sum_{i=0}^{m^{\epsilon_\ell}}\sum_{j=0}^{m^{v_\ell}}\Big((\rho_\ell\epsilon_\ell)_n e_n[(K_m)_P\phi_i^{\epsilon_\ell}-(K_m)_N\phi_{i,n}^{\epsilon_\ell}]\\
&+[\xi_n^{\epsilon_\ell}(\rho_\ell)_N\phi_{i,n}^{\epsilon_\ell}+\bar{\xi}_n^{\epsilon_\ell}(\rho_\ell)_P\phi_i^{\epsilon_\ell}]\phi_{j,n}^{v_\ell}\alpha_j^{v_\ell}\Big)\Delta y+\\
&+\sum_{i=0}^{m^{\epsilon_\ell}}\sum_{j=0}^{m^{v_\ell}}\Big((\rho_\ell\epsilon_\ell)_s e_s[(K_m)_P\phi_i^{\epsilon_\ell}-(K_m)_S\phi_{i,s}^{\epsilon_\ell}]\\
&+[\bar{\xi}_s^{\epsilon_\ell}(\rho_\ell)_S\phi_{i,s}^{\epsilon_\ell}+\xi_S^{\epsilon_\ell}(\rho_\ell)_P\phi_i^{\epsilon_\ell}]\phi_{j,s}^{v_\ell}\alpha_j^{v_\ell}\Big)\Delta x-\\
&-(\rho_\ell\epsilon_\ell)_e(u_\ell)_e\Delta y+(\rho_\ell\epsilon_\ell)_w(u_\ell)_w\Delta y-(\rho_\ell\epsilon_\ell)_n(v_\ell)_n\Delta y+(\rho_\ell\epsilon_\ell)_s(v_\ell)_s\Delta y-\\
&-[(\rho_\ell\epsilon_\ell)_P-(\rho_\ell\epsilon_\ell)_P^o]\frac{\Delta V}{\Delta t}
\end{aligned} \tag{4.19}$$

Equation (4.18) can be rearranged to generate a system of linear equations:

$$a_p^\epsilon\Big(\sum_{i=1}^{m^\epsilon}\alpha_i^{\epsilon^{'}}\phi_i^\epsilon,\phi_k^\epsilon\Big)_p-\sum_{nb}a_{nb}^\epsilon\Big(\sum_{i=1}^{m^\epsilon}\alpha_i^{\epsilon^{'}}\phi_i^\epsilon,\phi_k^\epsilon\Big)_{nb}=\Big(b_p^\epsilon,\phi_k^\epsilon\Big). \tag{4.20}$$

The linear system can be written in a more compact form:

$$\overline{A}^{\epsilon}\alpha^{\epsilon'} = \overline{B}^{\epsilon}, \tag{4.21}$$

where

$$\overline{A}^{\epsilon}_{ik} = a^{\epsilon}_{p}\Big(\sum_{i=1}^{m^{\epsilon}} \phi^{\epsilon}_{i}, \phi^{\epsilon}_{k}\Big)_{p} - \sum_{nb} a^{\epsilon}_{nb}\Big(\sum_{i=1}^{m^{\epsilon}} \phi^{\epsilon}_{i}, \phi^{\epsilon}_{k}\Big)_{nb} \tag{4.22}$$

$$\overline{B}^{\epsilon}_{i} = \Big(b^{\epsilon}_{p}, \phi^{\epsilon}_{k}\Big). \tag{4.23}$$

The dimensions of $\overline{A}_{ij}$ and $\overline{B}_{i}$ are m^{ϵ} x m^{ϵ} and m^{ϵ} x 1 respectively. The parameter m^{ϵ} is the number of solid volume fraction modes used in the ROM calculation. Only m^{ϵ} equations must be solved for the projected SVFCE given in (4.20). ODEx uses a time marching algorithm which is similar to the algorithm used by MFIX. An outline of the ODEx solution algorithm is given below:

- Using the time coefficients from the previous iteration or those read in from the initial condition file for the first time step, the field variables u_g, v_g, u_s, v_s, p_g and ϵ_g are reconstructed. For compressible flows, the density ρ_g is calculated using the ideal gas law with constant gas temperature.

- The linear system of equations (4.11) and (4.12) are solved to obtain the tentative values of $\alpha_i^{u_g}(t)$, $i \in [1, m^{u_g}]$, $\alpha_i^{v_g}(t)$, $i \in [1, m^{v_g}]$, $\alpha_i^{u_s}(t)$, $i \in [1, m^{u_s}]$ and $\alpha_i^{v_s}(t)$, $i \in [1, m^{v_s}]$. The values are called tentative because they are calculated based on the previous pressure field and will be corrected by the gas pressure correction and the solids volume fraction correction respectively.

- The linear system of equations (4.13) is solved to obtain $\alpha_i^{p'_g}(t)$, $i \in [1, m^{p_g}]$, the time coefficients of the gas pressure correction.

- The time coefficients of p_g, u_g and v_g are corrected using the new value of $\alpha_i^{p'_g}(t)$.

- The linear system of equations (4.34) is solved to obtain $\alpha_i^{\epsilon_s}{}'(t)$, $i \in [1, m^{\epsilon_s}]$, the time coefficients of the solids volume fraction correction. The void fraction, ϵ_g is calculated using $\epsilon_g = 1 - \epsilon_s$.

- The time coefficients of u_s, v_s, and ϵ_g are corrected using the new value of $\alpha_i^{\epsilon_s}{}'(t)$.

- Check the convergence. If the solution is converged, ODEx advances to the next time step.

The input data for ODEx are the basis functions of the gas and solids velocities, the gas pressure and the void fraction. The solutions of ODEx are $\alpha_i^{u_g}$, $i \in [1, m^{u_g}]$, $\alpha_i^{v_g}$, $i \in [1, m^{v_g}]$, $\alpha_i^{u_s}$, $i \in [1, m^{u_s}]$, $\alpha_i^{v_s}$, $i \in [1, m^{v_s}]$, $\alpha_i^{p_g}$, $i \in [1, m^{p_g}]$ and $\alpha_i^{\epsilon_g}$, $i \in [1, m^{\epsilon_g}]$. An example input file for ODEx can be found in Appendix B.

2. Reduced-Order Model Based on Proper Orthogonal Decomposition for the Scalar Energy Equations.

For a two-dimensional, non-isothermal, multiphase flow, MFIX solves the discretized x- and y-momentum equations for the gas and solids phases, the gas pressure correction equation, the solids volume fraction correction equation and the scalar gas and solids energy equations. Therefore there are eight unknowns per cell in MFIX. These variables are the gas and solids velocity components, u_g, v_g, u_s and v_s, the gas pressure p_g, the void fraction ϵ_g and the gas and solids temperature values T_g and T_s. For each of these field variables a database composed of snapshots has been generated and the POD basis functions $\phi_i^{u_g}$, $\phi_i^{v_g}$, $\phi_i^{u_s}$, $\phi_i^{v_s}$, $\phi_i^{p_g}$, $\phi_i^{\epsilon_g}$, $\phi_i^{t_g}$, and $\phi_i^{t_s}$ have been extracted from the data set.

a. ODEt

The projection of the scalar energy equation onto the temperature basis functions is derived below. First, the scalar energy equation for temperature can be written as [18, p. 54]

$$a_P(T_\ell)_P = \sum_{nb} a_{nb}(T_\ell)_{nb} + b_P \tag{4.24}$$

where ℓ denotes either the gas or solids phase (g or s). Next, equations (3.4e) and (3.4f) are substituted into (4.24),

$$-\sum_{nb}(a_m^T)_{nb}(T_\ell)_P - \frac{(\rho_\ell)C_{p\ell}\Delta V}{\triangle t}(T_\ell)_p = -\sum_{nb} a_{nb}(T_\ell)_{nb} - \frac{(\rho_\ell)C_{p\ell}\Delta V}{\triangle t}(T_\ell^o)_p + S_\ell^T \tag{4.25}$$

the source terms present in the new expression are combined into the term S_ℓ^T for convenience.

Equation (4.25) can be rearranged as

$$-\rho_\ell C_{p\ell}\Delta V\frac{(T_\ell)_p - (T_\ell^o)_p}{\triangle t} = \sum_{nb} a_{nb}((T_\ell)_{nb} - (T_\ell)_p) + S_\ell^T. \tag{4.26}$$

Replacing $\frac{(T_\ell)_p - (T_\ell^o)_p}{\triangle t}$ with $\frac{\partial T_\ell}{\partial t}$ yields,

$$-\rho_\ell C_{p\ell}\Delta V\frac{\partial T_\ell}{\partial t} = \sum_{nb} a_{nb}((T_\ell)_{nb} - (T_\ell)_p) + S_\ell^T. \tag{4.27}$$

The temperature T_ℓ is approximated using the POD basis functions and time coefficients in the following equation:

$$T_\ell(x,t) \cong \phi_0^{T_\ell}(x) + \sum_{i=1}^{m^{T_\ell}} \alpha_i^{T_\ell}(t)\phi_i^{T_\ell}(x). \tag{4.28}$$

Substituting (4.28) into (4.27) and dropping the x and t notation yields

$$-\rho_\ell C_{p\ell}\Delta V\sum_{i=1}^{m^{T_\ell}} \dot{\alpha}_i^{T_\ell}\phi_i^{T_\ell} = \sum_{nb} a_{nb}\sum_{i=1}^{m^{T_\ell}}(\phi_{i,nb}^{T_\ell} - \phi_i^{T_\ell})\alpha_i^{T_\ell} + S_\ell^T. \tag{4.29}$$

Substituting for the a coefficient terms given in Equations (3.4a - 3.4d) yields

$$\begin{aligned}
&\rho\Delta V \cdot \sum_{i=1}^{m^{T_\ell}} \dot{\alpha}_i^{T_\ell}\phi_i^{T_\ell} = \\
&-\sum_{i=0}^{m^{T_\ell}}\sum_{j=0}^{m^u} \frac{(\xi^{T_\ell})_e}{2}(\epsilon_\ell\rho_\ell)_E(C_{p\ell}+C_{p\ell E})\Delta y\phi_{j,e}^{u_\ell}(\phi_{i,e}^{T_\ell}-\phi_i^{T_\ell})\alpha_i^{T_\ell}\alpha_j^{u_\ell} + \\
&+\sum_{i=0}^{m^{T_\ell}}\sum_{j=0}^{m^u} \frac{(\bar{\xi}^{T_\ell})_w}{2}(\epsilon_\ell\rho_\ell)_W(C_{p\ell}+C_{p\ell E})\Delta y\phi_{j,w}^{u_\ell}(\phi_{i,w}^{T_\ell}-\phi_i^{T_\ell})\alpha_i^{T_\ell}\alpha_j^{u_\ell} - \\
&-\sum_{i=0}^{m^{T_\ell}}\sum_{j=0}^{m^u} \frac{(\xi^{T_\ell})_n}{2}(\epsilon_\ell\rho_\ell)_N(C_{p\ell}+C_{p\ell N})\Delta x\phi_{j,n}^{v_\ell}(\phi_{i,n}^{T_\ell}-\phi_i^{T_\ell})\alpha_i^{T_\ell}\alpha_j^{v_\ell} + \\
&+\sum_{i=0}^{m^{T_\ell}}\sum_{j=0}^{m^v} \frac{(\bar{\xi}^{T_\ell})_s}{2}(\epsilon_\ell\rho_\ell)_S(C_{p\ell}+C_{p\ell N})\Delta x\phi_{j,s}^{v_\ell}(\phi_{i,s}^{T_\ell}-\phi_i^{T_\ell})\alpha_i^{T_\ell}\alpha_j^{v_\ell} + \\
&+\sum_{i=0}^{m^{T_\ell}}(K_\ell)_e\Delta y(\phi_{i,e}^{T_\ell}-\phi_i^{T_\ell})\alpha_i^{T_\ell} + \sum_{i=0}^{m^{T_\ell}}(K_\ell)_w\Delta y(\phi_{i,w}^{T_\ell}-\phi_i^{T_\ell})\alpha_i^{T_\ell} + \\
&+\sum_{i=0}^{m^{T_\ell}}(K_\ell)_n\Delta x(\phi_{i,n}^{T_\ell}-\phi_i^{T_\ell})\alpha_i^{T_\ell} + \sum_{i=0}^{m^{T_\ell}}(K_\ell)_s\Delta x(\phi_{i,s}^{T_\ell}-\phi_i^{T_\ell})\alpha_i^{T_\ell} + \\
&+S_\ell^T,
\end{aligned} \tag{4.30}$$

For two-dimensional flows, $A_E = A_W = \Delta y$ and $A_N = A_S = \Delta x$. Equation (4.30) is the finite volume discretized scalar energy equation approximated using proper orthogonal decomposition. The S_ℓ^T terms are computed based on the previous time step values of the gas and solids temperatures. Next, Eq. (4.30) is projected onto the

basis functions $\phi_k^{T_\ell}$, where $k \in [1, m^{T_\ell}]$

$$
\begin{aligned}
&\rho\Delta V \cdot \sum_{i=1}^{m^{T_\ell}} \dot{\alpha}_i^{T_\ell}(\phi_i^{T_\ell}, \phi_k^{T_\ell}) = \\
&-\sum_{i=0}^{m^{T_\ell}}\sum_{j=0}^{m^u} \frac{(\xi^{T_\ell})_e}{2}(\epsilon_\ell\rho_\ell)_E(C_{p\ell}+C_{p\ell E})\Delta y\phi_{j,e}^{u_\ell}(\phi_{i,e}^{T_\ell}-\phi_i^{T_\ell}), \phi_k^{T_\ell}\alpha_i^{T_\ell}\alpha_j^{u_\ell} + \\
&+\sum_{i=0}^{m^{T_\ell}}\sum_{j=0}^{m^u} \frac{(\bar{\xi}^{T_\ell})_w}{2}(\epsilon_\ell\rho_\ell)_W(C_{p\ell}+C_{p\ell E})\Delta y\phi_{j,w}^{u_\ell}(\phi_{i,w}^{T_\ell}-\phi_i^{T_\ell}), \phi_k^{T_\ell}\alpha_i^{T_\ell}\alpha_j^{u_\ell} - \\
&-\sum_{i=0}^{m^{T_\ell}}\sum_{j=0}^{m^u} \frac{(\xi^{T_\ell})_n}{2}(\epsilon_\ell\rho_\ell)_N(C_{p\ell}+C_{p\ell N})\Delta x\phi_{j,n}^{v_\ell}(\phi_{i,n}^{T_\ell}-\phi_i^{T_\ell}), \phi_k^{T_\ell}\alpha_i^{T_\ell}\alpha_j^{v_\ell} + \\
&+\sum_{i=0}^{m^{T_\ell}}\sum_{j=0}^{m^v} \frac{(\bar{\xi}^{T_\ell})_s}{2}(\epsilon_\ell\rho_\ell)_S(C_{p\ell}+C_{p\ell N})\Delta x\phi_{j,s}^{v_\ell}(\phi_{i,s}^{T_\ell}-\phi_i^{T_\ell}), \phi_k^{T_\ell}\alpha_i^{T_\ell}\alpha_j^{v_\ell} + \\
&+\sum_{i=0}^{m^{T_\ell}}(K_\ell)_e\Delta y((\phi_{i,e}^{T_\ell}-\phi_i^{T_\ell}), \phi_k^{T_\ell})\alpha_i^{T_\ell} + \sum_{i=0}^{m^{T_\ell}}(K_\ell)_w\Delta y((\phi_{i,w}^{T_\ell}-\phi_i^{T_\ell}), \phi_k^{T_\ell})\alpha_i^{T_\ell} + \\
&+\sum_{i=0}^{m^{T_\ell}}(K_\ell)_n\Delta x((\phi_{i,n}^{T_\ell}-\phi_i^{T_\ell}), \phi_k^{T_\ell})\alpha_i^{T_\ell} + \sum_{i=0}^{m^{T_\ell}}(K_\ell)_s\Delta x((\phi_{i,s}^{T_\ell}-\phi_i^{T_\ell}), \phi_k^{T_\ell})\alpha_i^{T_\ell} + \\
&+(S_\ell^T, \phi_k^{T_\ell}),
\end{aligned}
\tag{4.31}
$$

Equation (4.31) can be written as

$$
\check{\mathcal{A}}_{kk}^{T_\ell}\dot{\alpha}_k^{T_\ell} = \sum_{i=0}^{m^{T_\ell}}\sum_{j=0}^{m^u}\check{\mathcal{F}}_{kij}^{T_\ell}\alpha_i^{T_\ell}\alpha_j^{u_\ell} + \sum_{i=0}^{m^{T_\ell}}\sum_{j=0}^{m^v}\check{\mathcal{G}}_{kij}^{T_\ell}\alpha_i^{T_\ell}\alpha_j^{v_\ell} + \sum_{i=0}^{m^{T_\ell}}\check{\mathcal{H}}_{ki}^{T_\ell}\alpha_i^{T_\ell} + \check{\mathcal{S}}^{T_\ell}, \tag{4.32}
$$

where

$$
\check{\mathcal{A}}_{ij}^{T_\ell} = \delta_{ij}\cdot(\rho_p\phi_j^{T_\ell}\Delta V, \phi_i^{T_\ell}),
$$

$$
\check{\mathcal{F}}_{kij}^{T_\ell} = (-(\xi^{T_\ell}\epsilon_\ell\rho_\ell C_{p\ell})_e\Delta y\phi_{j,e}^{u_\ell}(\phi_{i,e}^{T_\ell}-\phi_i^{T_\ell}), \phi_k^{T_\ell}) + ((\bar{\xi}^{T_\ell}\epsilon_\ell\rho_\ell)_w(C_{p\ell})_e\Delta y\phi_{j,w}^{u_\ell}(\phi_{i,w}^{T_\ell}-\phi_i^{T_\ell}), \phi_k^{T_\ell})
$$

$$
\check{\mathcal{G}}_{kij}^{T_\ell} = (-(\xi^{T_\ell}\epsilon_\ell\rho_\ell C_{p\ell})_n\Delta x\phi_{j,e}^{v_\ell}(\phi_{i,e}^{T_\ell}-\phi_i^{T_\ell}), \phi_k^{T_\ell}) + ((\bar{\xi}^{T_\ell}\epsilon_\ell\rho_\ell)_s(C_{p\ell})_n\Delta x\phi_{j,w}^{v_\ell}(\phi_{i,w}^{T_\ell}-\phi_i^{T_\ell}), \phi_k^{T_\ell})
$$

$$\begin{aligned}\check{\mathcal{H}}_{li}^{T_\ell} &= \Big(\frac{(K_\ell)_e\Delta y}{\Delta x}(\phi_{i,e}^{T_\ell}-\phi_i^{T_\ell}),\phi_k^{T_\ell}\Big)+\Big(\frac{(K_\ell)_w\Delta y}{\Delta x}(\phi_{i,w}^{T_\ell}-\phi_i^{T_\ell}),\phi_k^{T_\ell}\Big)\\ &+\Big(\frac{(K_\ell)_n\Delta x}{\Delta y}(\phi_{i,n}^{T_\ell}-\phi_i^{T_\ell}),\phi_k^{T_\ell}\Big)+\Big(\frac{(K_\ell)_s\Delta x}{\Delta y}(\phi_{i,s}^{T_\ell}-\phi_i^{T_\ell}),\phi_k^{T_\ell}\Big),\end{aligned}$$

$$\check{\mathcal{S}}_k^{T_\ell}=(S^{T_\ell},\phi_k^{T_\ell}).$$

In Eq. (4.32) the original energy balance equations have been reduced to a set of m^{T_ℓ} ODEs. Equation (4.32) can be rewritten in a simpler form in terms of coefficients $a_p^{T_\ell}$, $a_{nb}^{T_\ell}$, $b_p^{T_\ell}$ and unknowns, α^{T_ℓ}.

$$\overline{A}^{T_\ell}\alpha^{T_\ell}=\overline{B}^{T_\ell}. \tag{4.33}$$

where

$$\tilde{\mathcal{A}}_{ij}^{T_\ell}=\Big((a_p^{T_\ell}\phi_j^{T_\ell}-\sum_{nb}a_{nb}^{T_\ell}\phi_{j,nb}^{T_\ell}),\phi_i^{T_\ell}\Big),$$

$$\tilde{\mathcal{B}}_i^{T_\ell}=(b_p^{T_\ell},\phi_i^{T_\ell}).$$

Herein the dimensions of $\tilde{\mathcal{A}}^{T_\ell}$ and $\tilde{\mathcal{B}}^{T_\ell}$ are $m^{T_\ell}\times m^{T_\ell}$ and $m^{T_\ell}\times 1$, respectively. ODEt uses an iterative, time marching algorithm which is similar to the algorithm used by MFIX. An outline of the iterative loop of the ODEt solution algorithm is given below:

- Using the time coefficients from the previous iteration or those read in from the initial condition file for the first time step, the field variables u_g, v_g, u_s, v_s, p_g, ϵ_g, T_g and T_s are reconstructed. For compressible flows, the density ρ is calculated using the ideal gas law with the previous or reconstructed gas temperature. The physical properties ρ_g and ρ_s are calculated. The transport properties μ_g, μ_s and F_{gs} are calculated. The temperature dependent conductivity, K_m, and heat transfer coefficient, γ_{gs}, are calculated using the previous temperature field values.

- The linear systems of equations [20, p. 248, Eq. 30-31] are solved to obtain the

tentative values of $\alpha_i^{u_g}(t)$, $i \in [1, m^{u_g}]$, $\alpha_i^{v_g}(t)$, $i \in [1, m^{v_g}]$, $\alpha_i^{u_s}(t)$, $i \in [1, m^{u_s}]$ and $\alpha_i^{v_s}(t)$, $i \in [1, m^{v_s}]$ where m indicates the number of modes used. The values are called tentative because they are calculated based on the previous pressure field and will be corrected by the gas pressure correction and the solids volume fraction correction respectively.

- The linear system of equations [20, p. 249, Eq. 32] is solved to obtain $\alpha_i^{p'_g}(t)$, $i \in [1, m^{p_g}]$, the time coefficients of the gas pressure correction.
- The time coefficients of p_g, u_g and v_g are corrected using the new value of $\alpha_i^{p'_g}(t)$.
- The linear system of equations (4.34) is solved to obtain $\alpha_i^{\epsilon_s{}'}(t)$, $i \in [1, m^{\epsilon_s}]$, the time coefficients of the solids volume fraction correction. The void fraction, ϵ_g is calculated using $\epsilon_g = 1 - \epsilon_s$.
- The time coefficients of ϵ_s, u_s and v_s are corrected using the new $\alpha_i^{\epsilon_s{}'}(t)$ values.
- The linear systems of equations 4.33 are solved to obtain $(\alpha_i^{T_g})(t)$, $i \in [1, m^{T_g}]$ and $\alpha_i^{T_s}(t)$, $i \in [1, m^{T_s}]$, the time coefficients of the gas and solids phase temperature.
- Check the convergence. If the solution is converged, ODEt advances to the next time step.

The input data for ODEt are the basis functions extracted from the database of eight field variables generated using MFIX. The eight field variables are gas and solids velocities, gas pressure, void fraction and gas and solids temperatures. The solutions of ODEt are $\alpha_i^{u_g}$, $i \in [1, m^{u_g}]$, $\alpha_i^{v_g}$, $i \in [1, m^{v_g}]$, $\alpha_i^{u_s}$, $i \in [1, m^{u_s}]$, $\alpha_i^{v_s}$, $i \in [1, m^{v_s}]$, $(\alpha_i^{p_g})'$, $i \in [1, m^{p_g}]$, $(\alpha_i^{\epsilon_g})'$, $i \in [1, m^{\epsilon_g}]$, $\alpha_i^{T_g}(t)$, $i \in [1, m^{T_g}]$ and $\alpha_i^{T_s}(t)$, $i \in [1, m^{T_s}]$. An example input file for ODEt can be found in Appendix C. Table II shows a list

of the most important subroutines created for the POD-based reduced-order model, ODEt. The routines in Table II are especially important because they are used to perform the projection of the energy balance equations onto the temperature basis functions, solve the linear system of equations and reconstruct the temperature values from the computed time coefficients, α^{T_ℓ}. Table III shows a list of the most important subroutines modified for ODEt. Most of these routines were modified from the July 2006 version of MFIX. Most of the modifications involved adding new terms for the POD algorithm and converting the MFIX three-dimensional, multiple-phase routines into simpler two-dimensional, two-phase routines for ODEt. Table IV shows a list of support subroutines modified for ODEt. The routines in Table IV are called support routines because they are not directly used for calculation. These subroutines are used for reading in data and declaring new variables.

Table II. Most important routines created for ODEt.

Subroutine	New Subroutine Description
calc_ab_phi.f	Projects the energy equation system onto the basis functions extracted from the temperature snapshots
phi_phi_prod.f	Calculates the constant basis function products outside the iteration loop
reconstruct_t_m.f	Reconstructs the gas and solids phase temperatures from the basis functions and time coefficients
solve_lin_eq.f	Replaces original BICGS solver with LU decomposition solver

Table III. Most important routines modified for ODEt.

Subroutine	Modification
allocate_arrays.f	Defined dynamic allocation size for the basis function arrays, time coefficients, constant basis function products and all new energy equation parameters not defined in ODEx.
calc_mu_g.f	Implemented temperature dependent μ_g calculation
calc_k_g/s.f	Implemented 2-D, 2-phase version of routines
calc_gama	Implemented 2-D, 2-phase version of routine
conv_dif_phi.f	Implemented 2-D, 2-phase version of routine
bc_phi.f	Implemented no slip wall condition
iterate.f	Added write out operations for α^{T_ℓ}
physical_prop.f	Sets $C_{p\ell}$, $\gamma_{R\ell}$, and $\Delta H_{r\ell}$ from input data
source_phi.f	Implemented 2-D, 2-phase version of routine
solve_energy_eq.f	Modified the call line for the linear equation solver and calc_resid_s to solve for the time coefficient of temperature, α^{T_ℓ}. Added the call line for the new calc_ab_phi routine. Added error flag to identify if $T < 0$.
time_march.f	Added lines to open and write out computed time coefficients to the output files. Added call statements for reconstruct_t routines.

Table IV. Support routines modified for ODEt.

Subroutine	Modification
basis_mod.f	Added temperature basis function definitions
get_data.f	Added statement to print error flag for undefined input parameters
namelist.inc	Added number of modes for temperature parameters, n_Tm, and initial values of $C_{p\ell}$, $\gamma_{R\ell}$, and $\Delta H_{r\ell}$ to list of parameters to be read from the input file, *puv.dat*
param_mod.f	Added definitions for nT_g and nT_s, the integer number of modes used for temperature reconstruction
read_basis.f	Modified routine to read the temperature basis functions from the PODDEC generated basis function files
set_ic.f	Modified routine to read the first time coefficients for temperature from the PODDEC generated time coefficient files to generate an initial field solution
time_coe_mod.f	Added definitions for α_g^T and α_s^T, the gas and solids phase temperature time coefficients

b. Boundary Conditions for ODEx/ODEt

ODEx and ODEt use ghost cells to enforce a no-penetration boundary condition on the boundaries of the control volume. Boundary behavior is also implicitly communicated to the ROM by the modes extracted from the original MFIX data. The initial boundary values of the field variables in the ghost cells are determined by the reconstruction step. Within the sub-iteration the no-penetration and no-slip boundary conditions are specified by altering ghost cell coefficients in the assembled $\tilde{\mathcal{A}}$ matrix. This step is performed before projection of the $\tilde{\mathcal{A}}$ matrix onto the basis functions. The projected $\tilde{\mathcal{A}}$ matrix system is solved for the new time coefficients. The iterative calculation begins again with a new reconstruction of the field variables. The following two paragraphs explain the boundary condition specification procedure in greater detail. In the following paragraphs, ODEx and ODEt will be refered to as ODEx to shorten the naming notation.

The basic implementation of the no-penetration, no-slip boundary condition in ODEx was created using the assumption that the boundary conditions are constant. At the beginning of each time step the initial boundary values are set by the first reconstruction of the field variables using the time coefficients and basis functions. During iterative calculation steps the boundary conditions are enforced so that the flux between the ghost cells and the cells along the computational domain perimeter is zero. This is achieved by specifying the coefficients of the assembled matrices before projection onto the basis functions. For flow parallel to the walls the boundary condition specified at the internal face of the ghost cell is of equal magnitude but opposite direction of the adjacent face.

The basic implementation of the no-penetration, no slip boundary condition described above is not sufficient for time dependent boundary conditions. Imple-

mentation of time dependent boundary conditions in ODEx involved specifying the exact boundary condition used in the full model MFIX. As an example, a simulation was run in MFIX where the central jet velocity (v_g) along the lower boundary of the control volume varied by $12.6 + 3.15 * SIN(60 * t)cm/s$, where t is the physical time of the calculation. In ODEx the value of the central jet velocity (v_g) was also set to $12.6 + 3.15 * \sin(60 * t)cm/s$ for the ghost cells. The no-penetration, no-slip boundary condition is maintained along the perimeter of the computational domain in the sub-iterations exactly as described in the previous paragraph.

C. Summary

The first section in this chapter presented the general scheme for generating the POD-based ROMs. An example governing equation (4.1) was used to illustrate the basic steps used to generate POD-based ROMs. In general the order of the governing equations are reduced by (1) replacing governing PDEs with a system of ODEs and (2) reducing the number of equations from N spatial points to m selected modes for each time step. PODDEC is used to extract a basis for modal decomposition from a database of snapshots. The database of snapshots is used to benerate the auto-correlation matrix $R(\mathbf{x}, \mathbf{x}')$. This eigenvalue problem is solved for time independent orthonormal basis functions, ϕ_k, and time dependent time coefficients, α_k. The basis functions and time coefficients are used for modal decomposition of the field variable. The PDE is projected onto the basis functions to form a system of ODEs.

The second section in this chapter described two POD-based ROMs generated to model flow features in a fluidized bed. The first, ODEx, models isothermal transport phenomena in a fluidized bed. The second, ODEt, simulates non-isothermal heat transfer for convection in a fluidized bed.

The discretized solids volume fraction correction equation and the discretized energy balance equation were projected onto the basis functions ϕ^{ϵ_s} and ϕ^{T_ℓ}, respectively. Two systems of linear algebraic equations were obtained:

$$\tilde{\mathcal{A}}^{\epsilon_s}\alpha^{\epsilon'_s} = \tilde{\mathcal{B}}^{\epsilon_s}, \tag{4.34}$$

$$\tilde{\mathcal{A}}^{T_\ell}\alpha^{T_\ell} = \tilde{\mathcal{B}}^{T_\ell}, \tag{4.35}$$

The steps of the POD-ROM algorithm were presented. A description of subroutines that were modified or created for the reduced-order model were included. This was done to aid understanding of the numerical implementation of the solution algorithm. A description of the no-penetration and no-slip boundary conditions implemented in the ROM was presented. Results from the POD-based ROM developed in the section are presented in the next section.

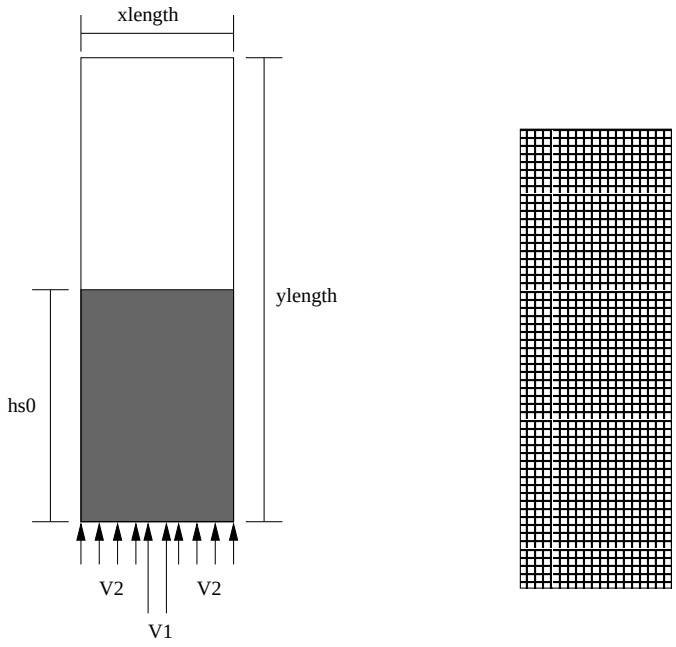
CHAPTER V

RESULTS

The purpose of this chapter is to present the results of the POD-based ROMs derived in the previous chapter. Four cases are investigated. Each case is a two-phase simulation consisting of one gas and one solids phase. The first case is an isothermal flow simulation. The second case is similar to the first except non-isothermal flow is modeled. The third case is an investigation into the ability to perform time extrapolation with the ROMs. The fourth case is an investigation of the ability of ODEx to simulate a periodic time dependent boundary condition.

A. Case I: Isothermal Flow

Case I is a multi-phase model and consists of one gas and one solids phase. The geometry, boundary conditions and computational domain of Case I are shown in Figures 5(a) and 5(b). The computational domain consists of a rectangular uniform grid. As described in the Physical Model chapter, Section B, air enters the through the lower boundary. A central jet with a velocity of 12.6 cm/s is surrounded on both sides by a lower velocity (1.0 cm/s) distributed gas inlet. The central jet inlet is 1.0 cm wide. The specific parameters of Case I are listed in Table V and are based on parameters used by Yuan.[9] The particle diameter is assumed to be uniform and constant. The gas and solids phases are each modeled as a single species and the flow is non-reacting.



(a) Geometry and boundary conditions

(b) Computational grid

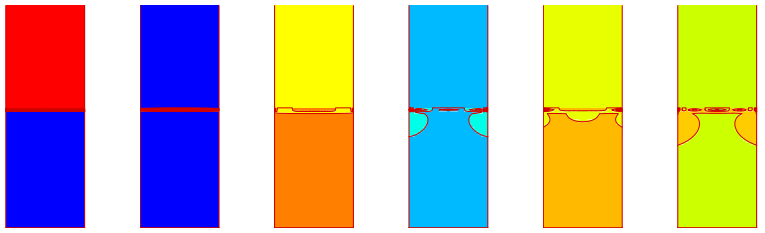
Fig. 5. Case I: geometry, boundary conditions, and computational grid.

Table V. Parameters of Case I.

Parameter	Description	Value
x_{length}	Length of the domain in x-direction	25.4cm
y_{length}	Length of the domain in y-direction	76.5cm
$imax$	Number of cells in x-direction	108
$jmax$	Number of cells in y-direction	124
v_1	Jet gas inflow velocity	12.6cm/s
v_2	Distributed gas inflow velocity	1.0cm/s
p_s	Static gas pressure at outlet	1.01×10^6g/(cm$\cdot s^2$)
T_{g0}	Gas temperature	297K
μ_{g0}	Gas viscosity	1.8×10^{-4}g/(cm$\cdot$ s)
t_{start}	Start time	0s
t_{stop}	Stop time	1s
$\triangle t$	Initial time step	1.0×10^{-4}s
ρ_{so}	Constant solids density	1.0g/cm^3
D_p	Solids particle diameter	0.5mm
h_{s0}	Initial packed bed height	38.25cm
ϵ_g^*	Packed bed void fraction	0.40

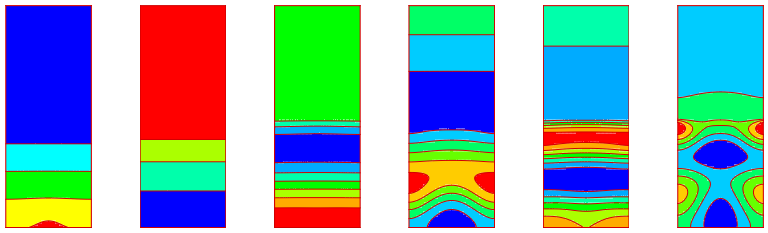
It is important to note that MFIX uses CGS units by default.

POD is applied to the database of snapshots computed by MFIX for six field variables modeled using the parameters in Table V. The first six POD basis functions of ϵ_g, p_g, u_g, v_g,u_s, and v_s are shown in Figures 6-12, respectively. ϕ_0 is the basis function that corresponds to the average mode.



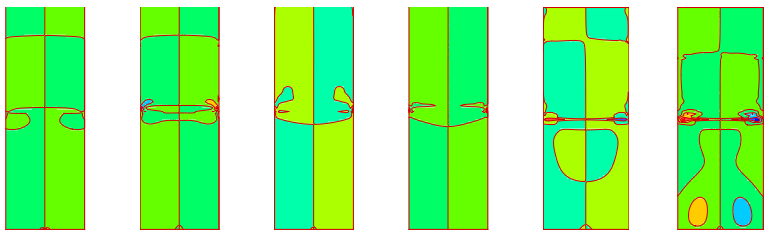
$\phi_0^{\epsilon_g}$ $\phi_1^{\epsilon_g}$ $\phi_2^{\epsilon_g}$ $\phi_3^{\epsilon_g}$ $\phi_4^{\epsilon_g}$ $\phi_5^{\epsilon_g}$

Fig. 6. Case I: first six basis functions of ϵ_g.



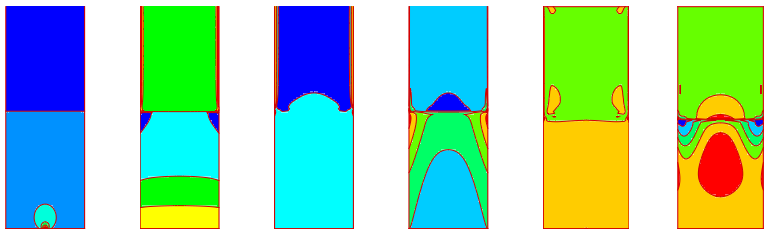
$\phi_0^{p_g}$ $\phi_1^{p_g}$ $\phi_2^{p_g}$ $\phi_3^{p_g}$ $\phi_4^{p_g}$ $\phi_5^{p_g}$

Fig. 7. Case I: first six basis functions of p_g.



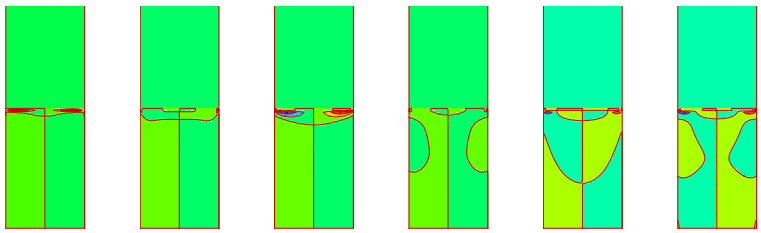
$\phi_0^{u_g}$ $\phi_1^{u_g}$ $\phi_2^{u_g}$ $\phi_3^{u_g}$ $\phi_4^{u_g}$ $\phi_5^{u_g}$

Fig. 8. Case I: first six basis functions of u_g.



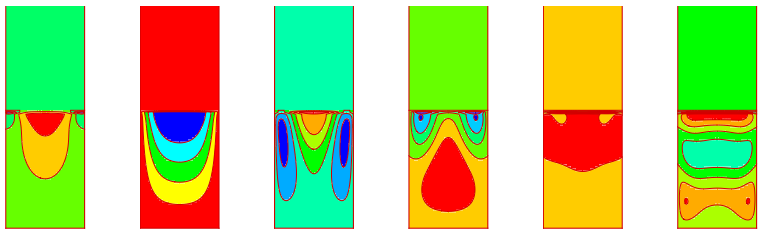
$\phi_0^{v_g}$ $\phi_1^{v_g}$ $\phi_2^{v_g}$ $\phi_3^{v_g}$ $\phi_4^{v_g}$ $\phi_5^{v_g}$

Fig. 9. Case I: first six basis functions of v_g.



$\phi_0^{u_s}$ $\phi_1^{u_s}$ $\phi_2^{u_s}$ $\phi_3^{u_s}$ $\phi_4^{u_s}$ $\phi_5^{u_s}$

Fig. 10. Case I: first six basis functions of u_s.



$\phi_0^{v_s}$ $\phi_1^{v_s}$ $\phi_2^{v_s}$ $\phi_3^{v_s}$ $\phi_4^{v_s}$ $\phi_5^{v_s}$

Fig. 11. Case I: first six basis functions of v_s.

Tables VI and VII show the percentage of the energy captured by each individual mode and the cumulative energy captured by the sum over the modes for each field variable.

Table VI. Energy variation for the gas pressure and gas velocities.

Number of Modes	p_g		u_g		v_g	
	Energy	Total	Energy	Total	Energy	Total
	[%]	[%]	[%]	[%]	[%]	[%]
1	99.938	99.938	90.102	90.102	85.018	85.018
2	0.589	99.997	6.274	96.377	13.646	98.664
3	0.024	99.999	2.914	99.291	0.986	99.650
4	0.004	99.999	0.305	99.596	0.248	99.898
5	0.000	99.999	0.220	99.815	0.074	99.972
6	0.000	99.999	0.120	99.993	0.013	99.985
7	0.000	99.999	0.026	99.996	0.008	99.993
8	0.000	99.999	0.017	99.997	0.004	99.997

Table VII. Energy variation for the solids velocities and void fraction.

Number of Modes	u_s		v_s		ϵ_g	
	Energy	Total	Energy	Total	Energy	Total
	[%]	[%]	[%]	[%]	[%]	[%]
1	80.644	80.644	96.559	96.559	68.597	68.597
2	15.373	96.017	2.018	98.577	27.541	96.138
3	1.818	97.835	1.069	99.647	3.691	99.829
4	1.252	99.086	0.203	99.850	0.112	99.941
5	0.495	99.582	0.121	99.971	0.043	99.983
6	0.223	99.804	0.018	99.989	0.014	99.998
7	0.098	99.902	0.007	99.996	0.001	99.999
8	0.068	99.970	0.002	99.999	0.000	99.999

1. Case I Results

For the test case chosen, the number of modes used and the energy spectrum captured by the chosen number of modes for each of the six state variables are shown in Table VIII.

Table VIII. Cumulative energy captured by the chosen number of modes.

Variable	Number of Modes	Symbol	Cumulative Energy [%]
p_g	2	N_{pg}	99.997
u_g	2	N_{ug}	96.377
v_g	5	N_{vg}	99.972
u_s	8	N_{us}	99.970
v_s	6	N_{vs}	99.989
ϵ_g	7	$N_{\epsilon g}$	99.999

The modes were chosen as a compromise between the accuracy and computational speed up of the reduced-order model. The selection of this mode combination was just one from a large number of mode combinations investigated. Due to the complex relationships between the six field variables modeled, discerning a distinct pattern between the number of modes used, the accuracy of the solution and the computational advantage was quite difficult. For example, it was discovered that decreasing the number of modes used to describe u_g improved the solution of v_s when the number of modes for the other variables remain fixed. This type of relationship is not intuitive and was only discovered by numerical experiments. The gas pressure and gas velocity variables can accurately be represented by only a few modes. Through

numerical testing, it was found that convergence of the solution was most sensitive to the number of pressure modes used. The solids velocities and void fraction parameters require more modes to capture because the solids flow features are more complex and have relatively small velocities when compared the gas velocities. For Case I, the maximum gas velocity in the y-direction was 12.6 cm/s while the maximum solids velocity in the y-direction is only 0.003 cm/s. Using the modes in Table VIII, it was possible to demonstrate that the new ODEx code was more computationally efficient than the previous POD, Hybrid_puv code. A summary of the computational time results for MFIX, Hybrid_puv and ODEx is shown in Table IX. The required compu-

Table IX. Summary of Case I CPU times for MFIX, Hybrid_puv and ODEx.

Code	Np_g	Nu_g	Nv_g	Nu_s	Nv_s	$N\epsilon_g$	t_{CPU} [s]
MFIX	-	-	-	-	-	-	9761.453
Hybrid_puv	2	2	5	8	6	-	3740.247
ODEx	2	2	5	8	6	7	894.317

tation time for MFIX was 9761.453 seconds, for Hybrid_puv 3740.247 seconds, and for ODEx 894.317 seconds. The POD code ODEx was 11 times faster than the full-order model code, MFIX, and 4 times faster than the previous POD code, Hybrid_puv. The results of the ODEx and Hybrid_puv codes are presented and compared with the solution of the full numerical model in the following sections.

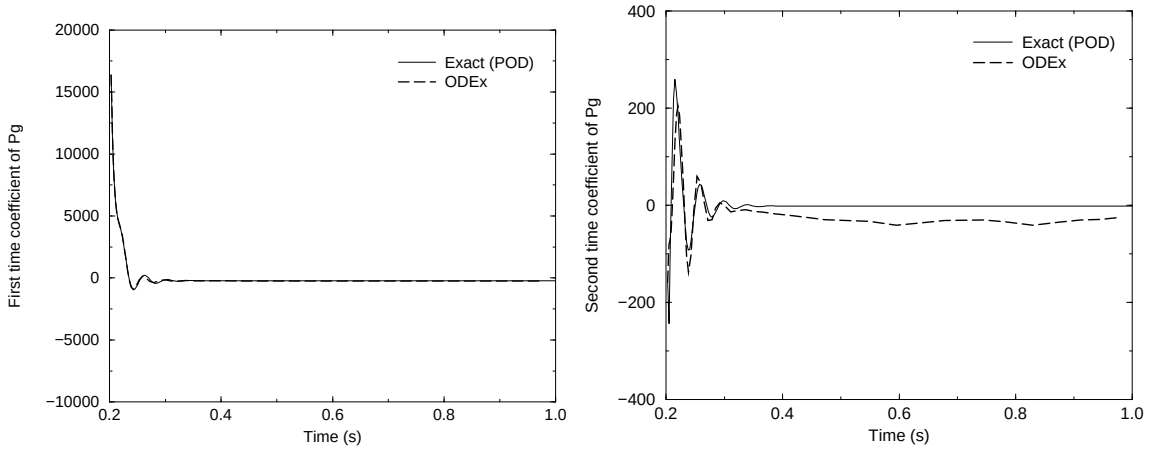


Fig. 12. Case I: The first two time coefficients of p_g using ODEx.

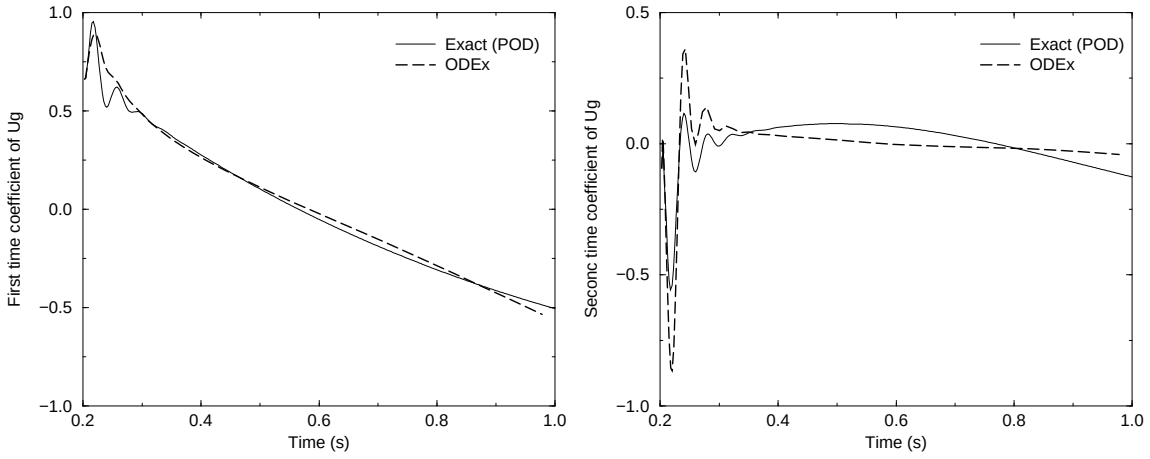


Fig. 13. Case I: The first two time coefficients of u_g using ODEx.

One method to assess the accuracy of the POD model is to compare its computed time coefficient values with those obtained by directly projecting the database of snapshots onto the basis functions. The latter procedure is used in the proper orthogonal decomposition of the snapshots and gives the 'exact solution' of the time coefficients. Comparisons of these two values are shown in Figures 12-17.

It can be seen that in most cases the time coefficients computed by ODEx match relatively well with the "exact" time coefficients. The one major exception to this trend can be found in the third and fourth time coefficients of v_g. The results of ODEx did not agree well with the directly projected time coefficients for these two modes. However, since the characteristics of v_g are dominated by the first two modes, these errors do not greatly affect the solution. Occasionally, errant modes such as these can be eliminated from the calculations because the fourth and fifth modes are relatively small when compared to the first and second. However, in this case using $Nv_g = 2$ instead of $Nv_g = 5$ required more iterations to achieve an accurate, converged solution and therefore was detrimental to the computation speed.

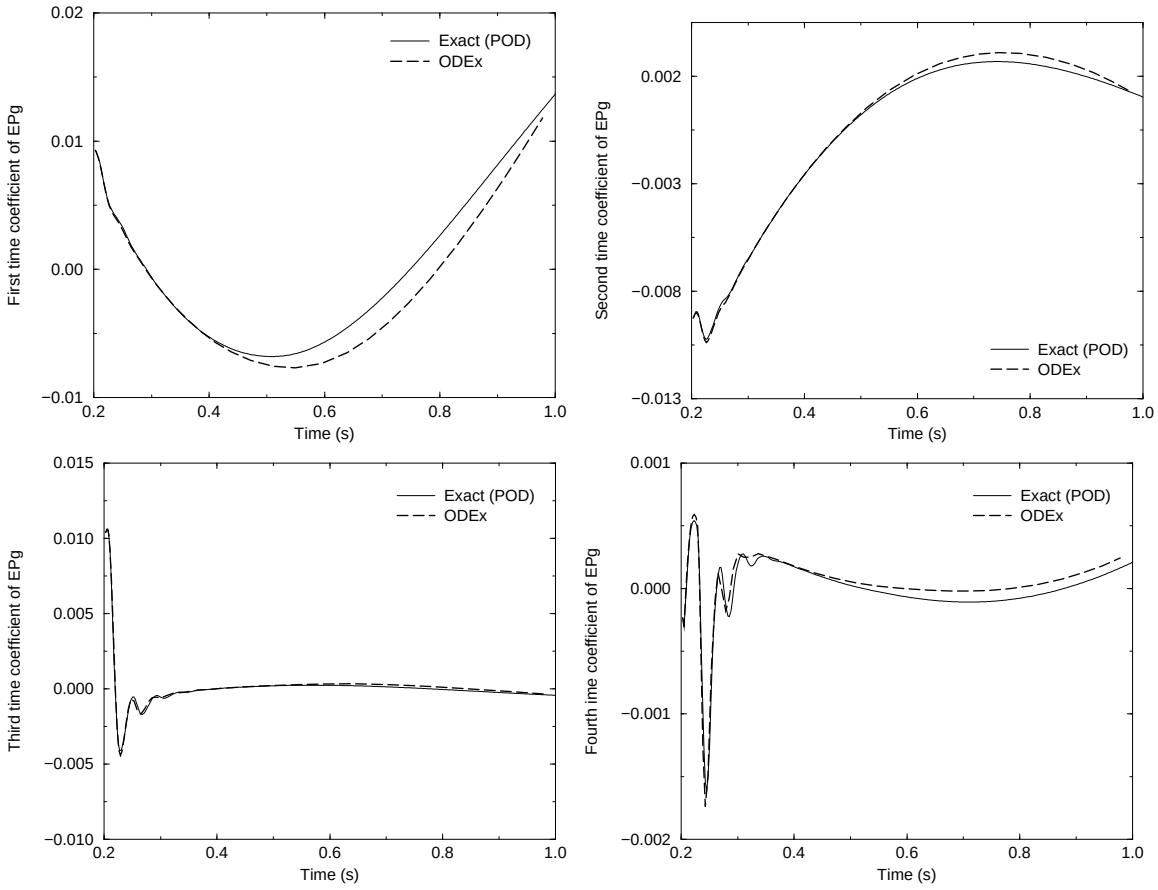


Fig. 14. Case I: The first four time coefficients of ϵ_g using ODEx.

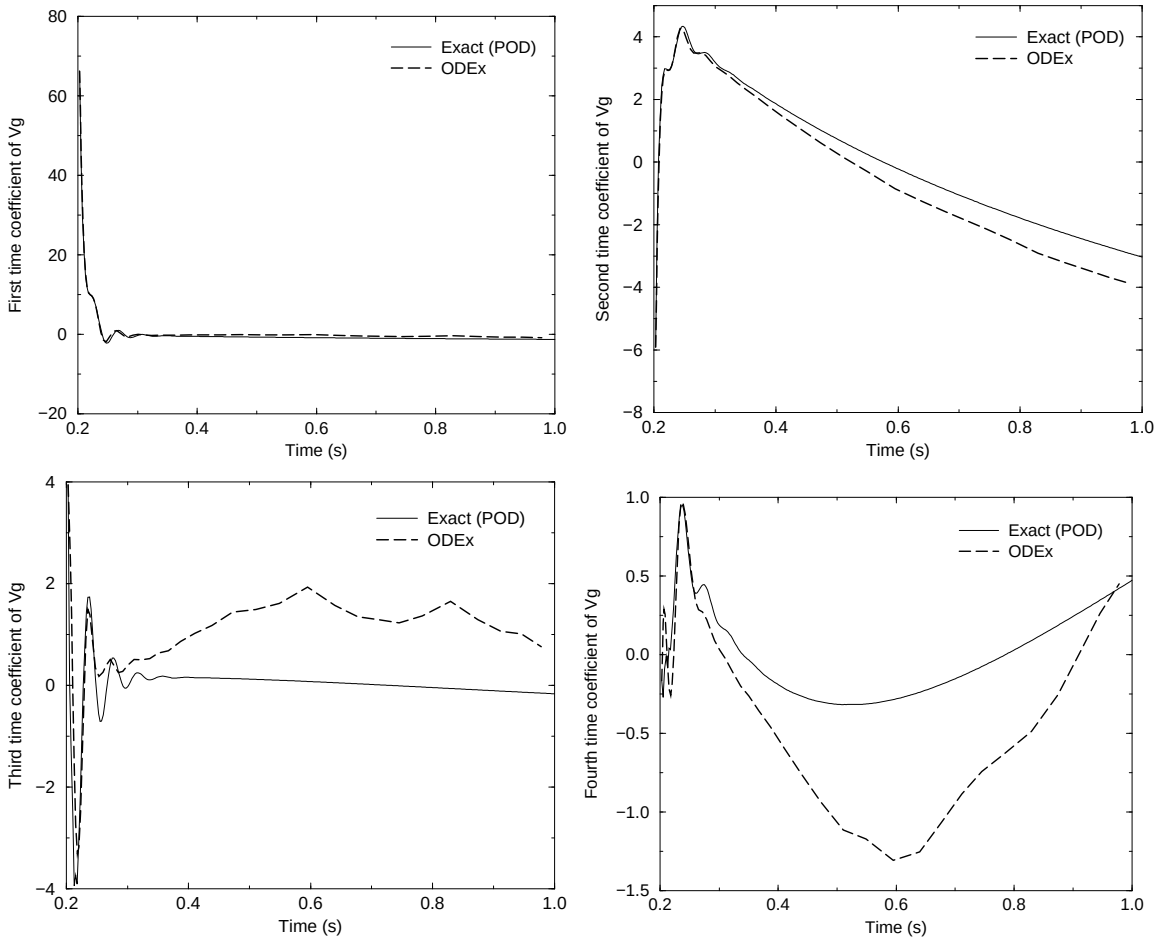


Fig. 15. Case I: The first four time coefficients of v_g using ODEx.

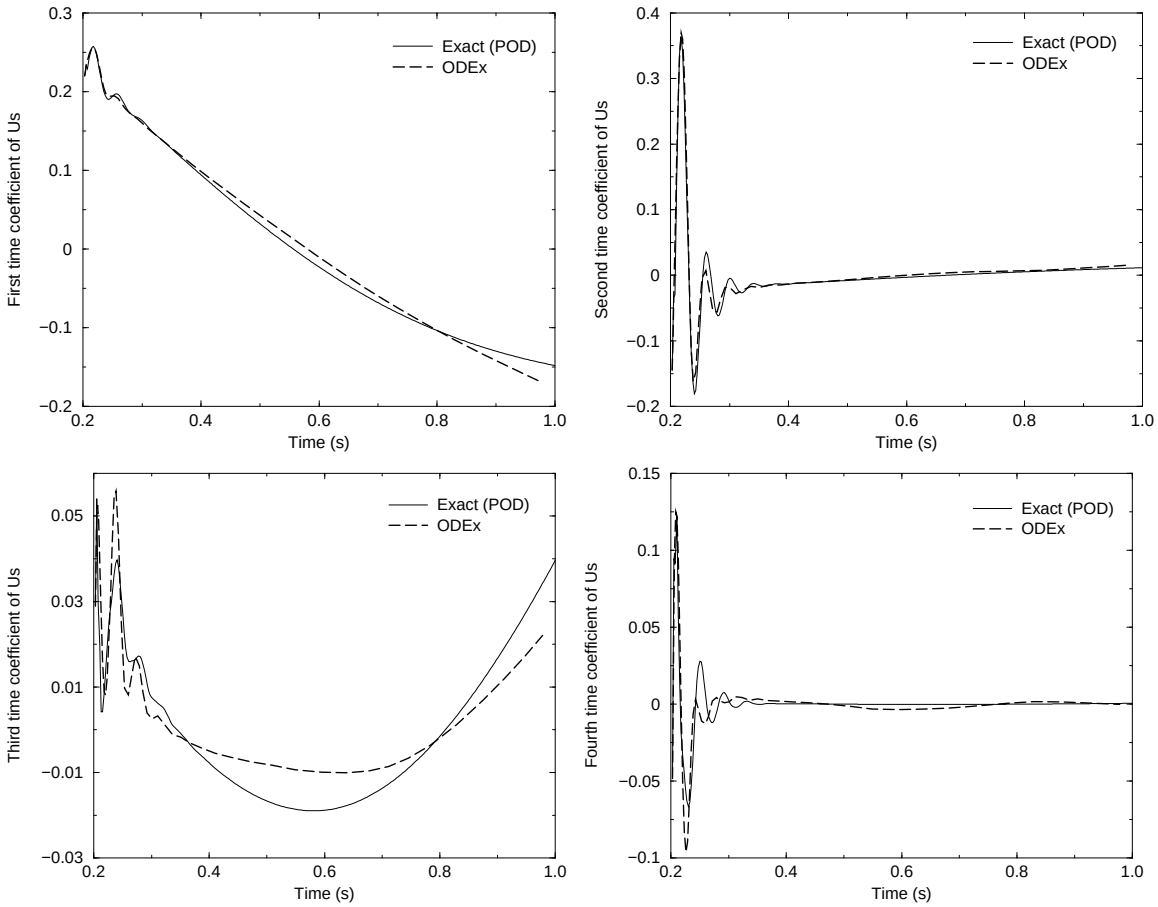


Fig. 16. Case I: The first four time coefficients of u_s using ODEx.

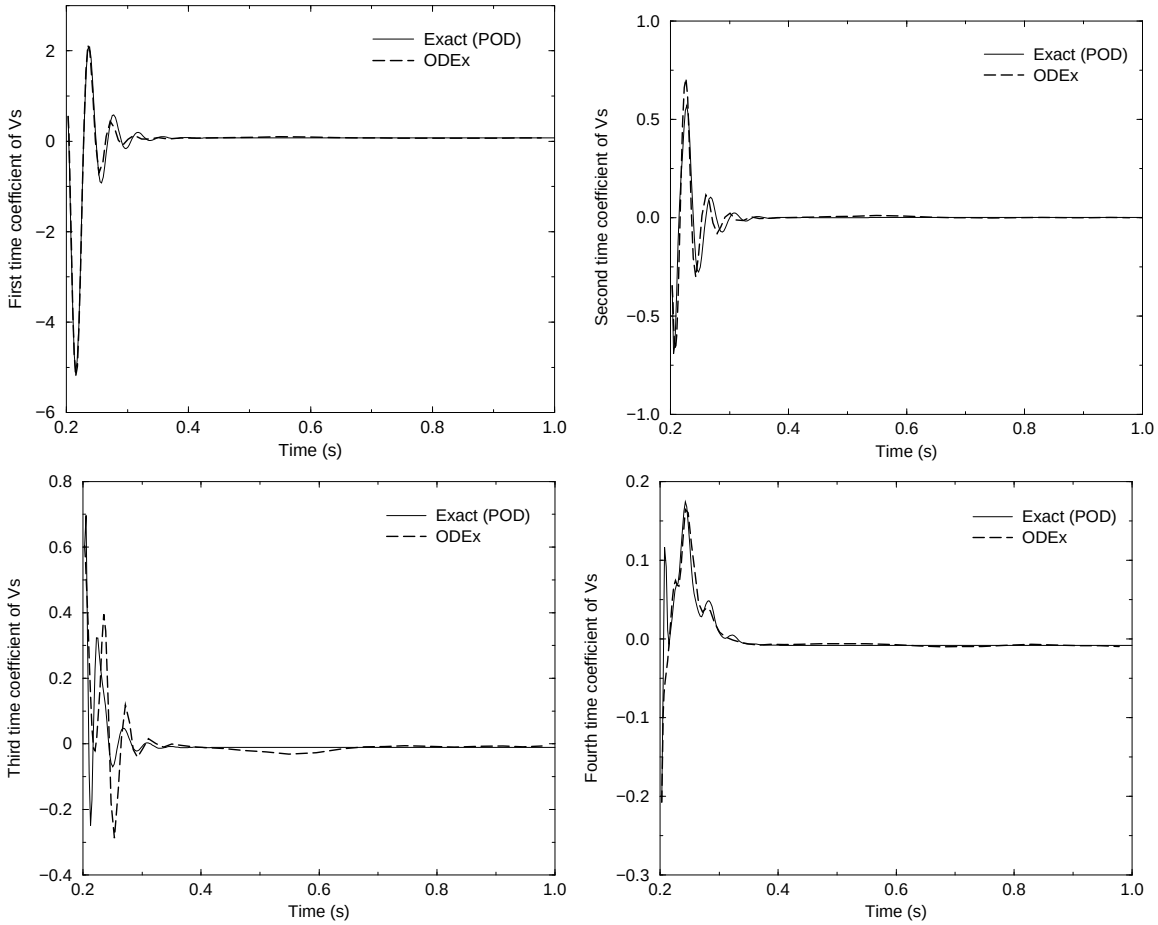


Fig. 17. Case I: The first four time coefficients of v_s using ODEx.

At the end of the computation, contour plots of the dependent field variables were created at time $t_{physical} = 1.0$ second. The numerical results at the final step of the calculation will accumulate the errors produced during the integration making these plotted flow fields the worst case results. For comparison, Case I was also tested using a hybrid POD model called Hybrid_puv. This POD model is referred to as a hybrid because the calculation of the void fraction was done by numerically solving the governing PDE rather than using a reduced-order model. Because of this, Hybrid_puv is able to more accurately compute the void fraction and solids volume fraction but the higher accuracy comes with the cost of reduced computational efficiency. The contour

plots in Figures 18-20 show the six state variables as computed by MFIX, Hybrid_puv and ODEx with the exception of the void fraction results for Hybrid_puv because the void fraction is computed using the same method as in MFIX. Each column of plots represent one state variable. The contour plots show good correlation between the flow features computed by the full numerical model MFIX and the reduced-order models. The one exception can be found in the y-direction solids velocity, v_s, computed by ODEx. It is shown in Figure 20 that ODEx correctly computes most of the v_s flow features but contains errors in the central region of the bed. This error is partially attributed to the small magnitude of the v_s velocities relative to the magnitude of the other velocities. The accuracy of v_s diminishes when fewer modes are used for any of the solids parameters in ODEx. The accuracy of the y-direction solids velocities is particularly sensitive to changes in the number of modes used for the void fraction. This dependence on additional state variables makes it difficult to achieve greater computational efficiency by simply reducing the number of modes used to represent the solids parameters. For example, Nu_s could easily be reduced from its current value of 8 to 3 modes and still give reasonably accurate results for the x-direction solids velocities. However, the solution for the y-direction solids velocities would become very poor. The ODEx model does achieve a one order of magnitude computational speed-up compared to MFIX and is capable of calculating qualitatively accurate results for Case I using the modes shown in Table VIII.

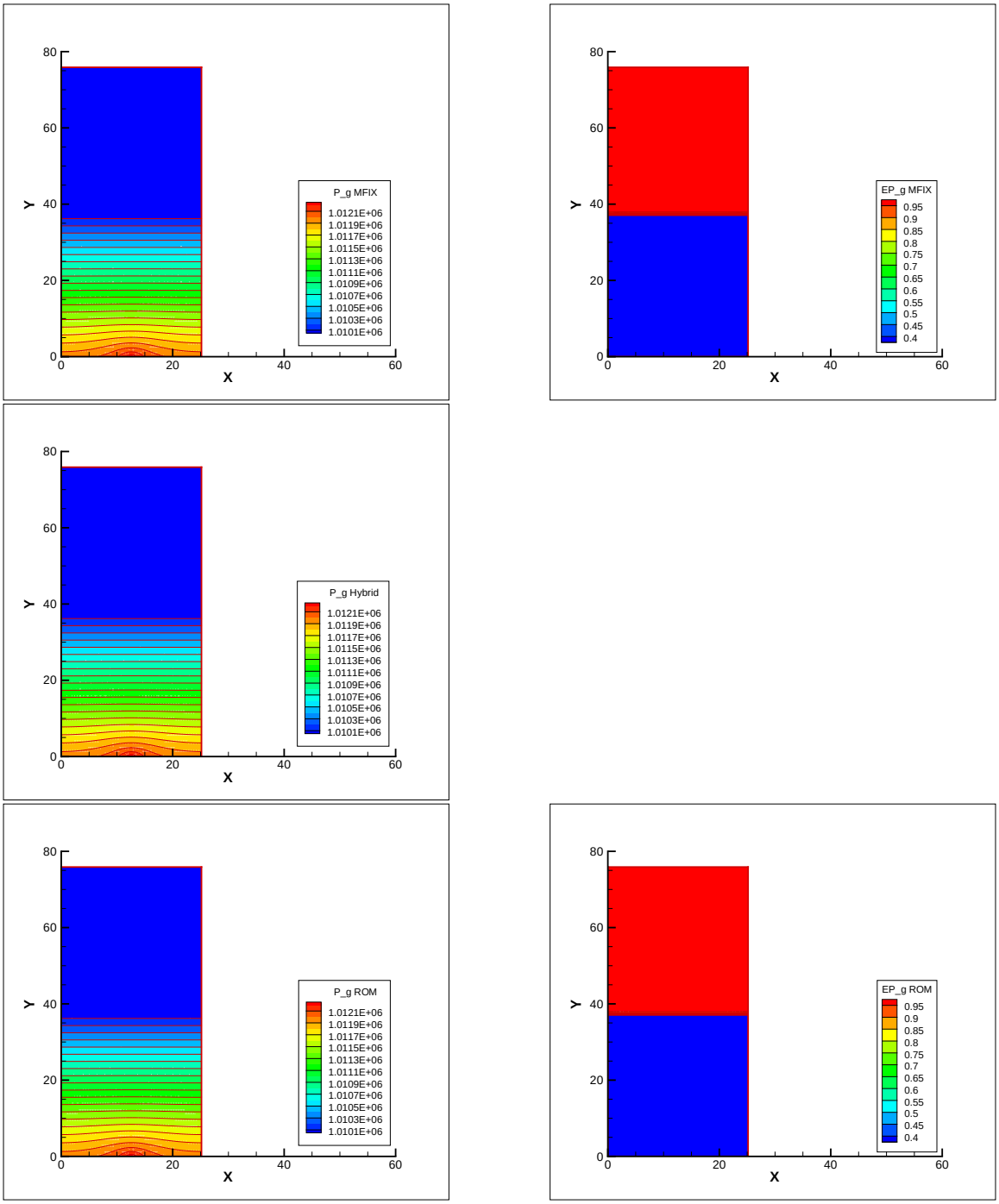


Fig. 18. Contour plots at $t_{physical} = 1.0$ seconds using MFIX, Hybrid and ODEx: gas pressure (p_g) and void fraction (ϵ_g).

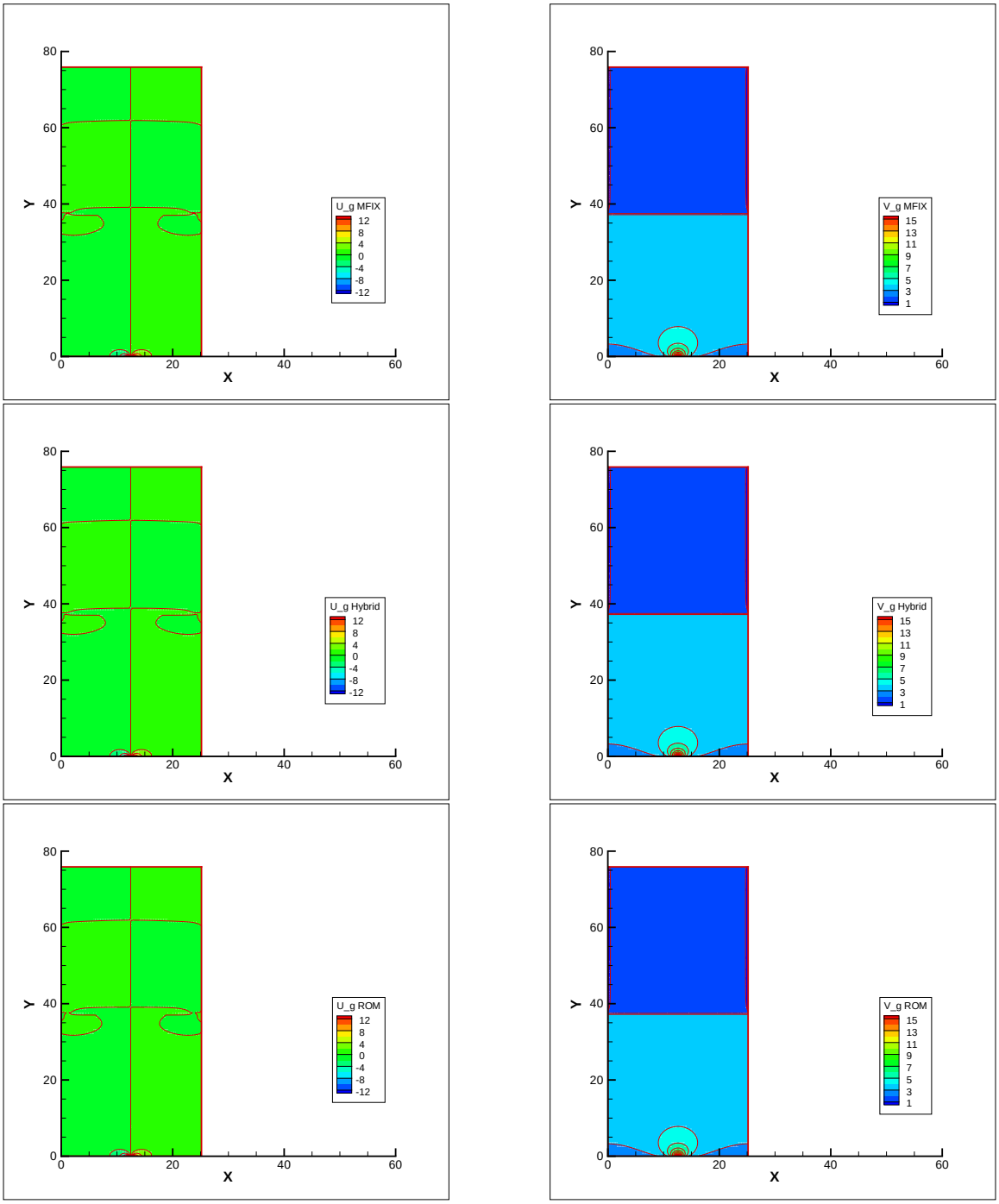


Fig. 19. Contour plots at $t_{physical} = 1.0$ second using MFIX, Hybrid and ODEx: u_g and v_g gas velocities.

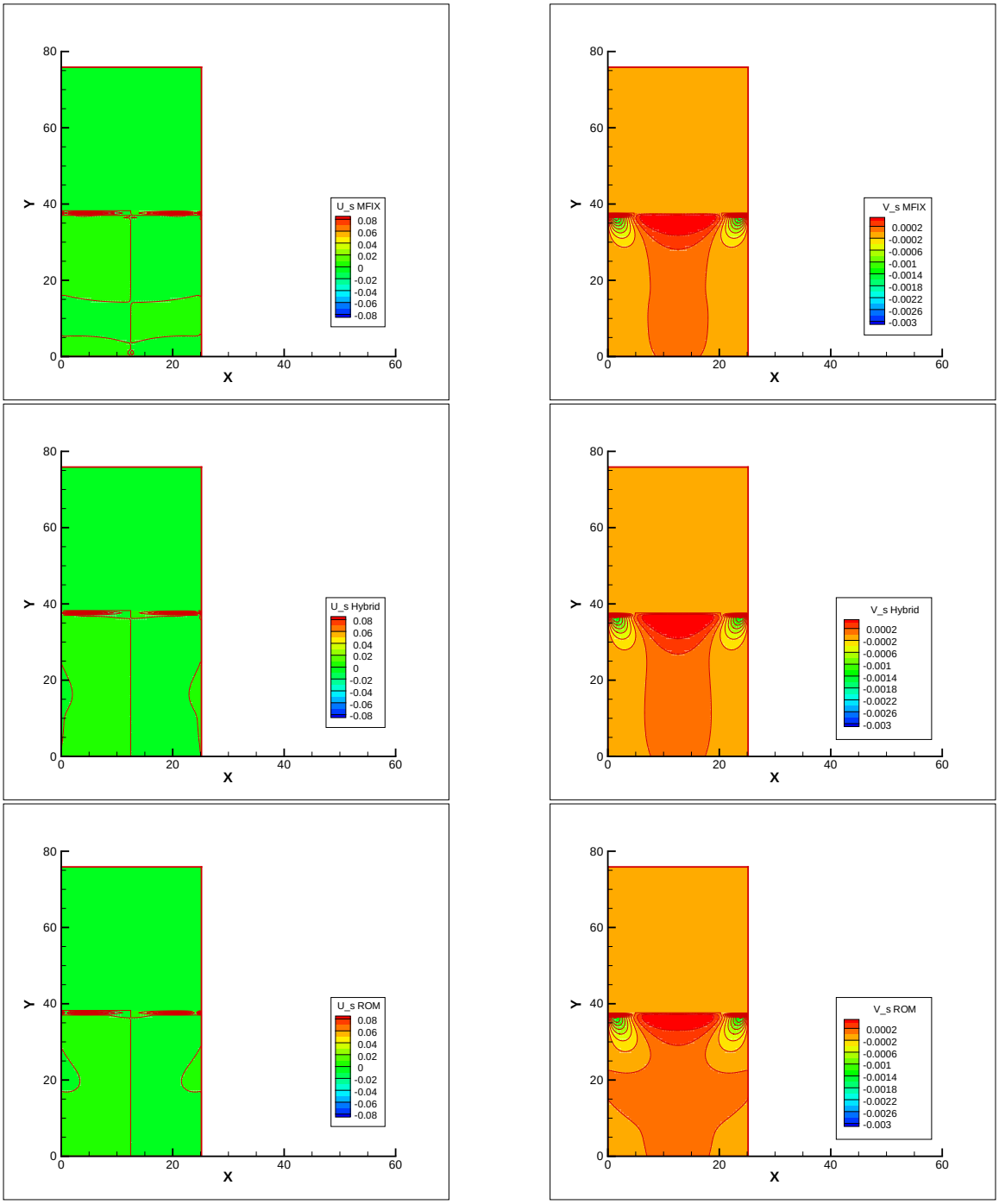


Fig. 20. Contour plots at $t_{physical} = 1.0$ second using MFIX, Hybrid and ODEx: u_s and v_s solids velocities.

B. Case II: Non-Isothermal Flow

Case II models a two-phase flow that consists of one gas phase and one solids phase. The boundary conditions of Case II are shown in Figure 21. The computational domain of Case II is identical to that of Case I shown in Figure 5(b). The boundary conditions are similar to those in Case I, however in Case II, the incoming gas has a higher temperature than the initial gas and solids in the computational domain. The specific parameters of Case II are listed in Table X. The values used for the initial specific heat coefficients C_{pg0} and C_{ps0} are those for air and ash at a temperature of 297 K respectively. In this case, radiation and internal heat sources are not modeled. Heat transfer is achieved only by convection and diffusion between hot gas entering the control volume and the gas and solids bed at an initially cooler temperature. As in Case I, the particle diameter is assumed to be uniform and constant. The gas and solids phases are modeled as a single species each and the flow is non-reacting.

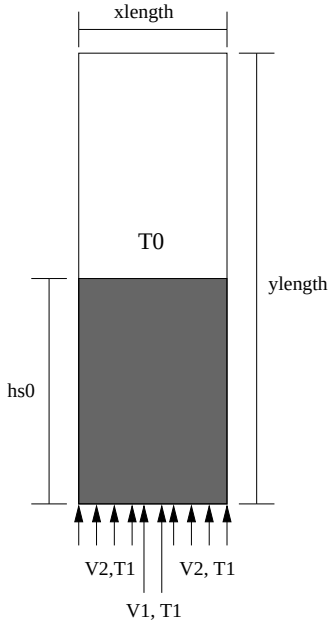


Fig. 21. Case II: geometry and boundary conditions.

Table X. Parameters of Case II.

Parameter	Description	Value
x_{length}	Length of the domain in x-direction	25.4cm
y_{length}	Length of the domain in y-direction	76.5cm
$imax$	Number of cells in x-direction	108
$jmax$	Number of cells in y-direction	124
v_1	Jet gas inflow velocity	12.6cm/s
v_2	Distributed gas inflow velocity	1.0cm/s
p_s	Static gas pressure at outlet	1.01×10^6g/(cm$\cdot s^2$)
T_{g0}	Initial gas temperature	297K
T_{s0}	Initial solids temperature	297K
T_{g1}	Inlet gas temperature	450K
μ_{g0}	Initial gas viscosity	1.8×10^{-4}g/(cm$\cdot$s)
t_{start}	Start time	0s
t_{stop}	Stop time	1.0s
$\triangle t$	Initial time step	1.0×10^{-4}s
ρ_{so}	Constant solids density	1.0g/cm^3
D_p	Solids particle diameter	0.5mm
h_{s0}	Initial packed bed height	38.25cm
C_{pg0}	Initial gas phase specific heat	0.25 cal/gK
C_{ps0}	Initial solids phase specific heat	0.310713 cal/gK
ϵ_g^*	Packed bed void fraction	0.40

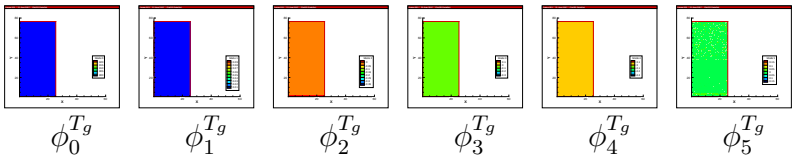


Fig. 22. Case II: first six basis functions of T_g.

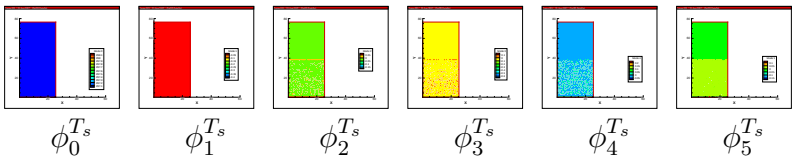


Fig. 23. Case II: first six basis functions of T_s.

POD is applied to the database of output computed by MFIX for the eight field variables modeled using the parameters in Table X. The first six POD basis functions of T_g and T_s are shown in Figures 22 and 23. The first six POD basis functions of ϵ_g, p_g, u_g, v_g,u_s, and v_s are shown in Figures 6-12 respectively in the previous section. They are not shown here because variation between the basis functions of the isothermal case and non-isothermal is qualitatively imperceivable in the contour plots. The six other field variables are not completely decoupled from the the scalar energy equations and temperature variations. The gas density, gas pressure, and gas viscosity are all affected by the gas temperature. However, for the temperature changes present in the flow, the change in the other flow features is qualitatively small.

At the end of the computation, the reconstructed variables were outputted to produce contour plots of the state variables at time $t_{physical} = 1.0$ second. Figures 24 and 25 show zoomed in views of the temperature distribution computed by MFIX and ODEt near the jet inlet.

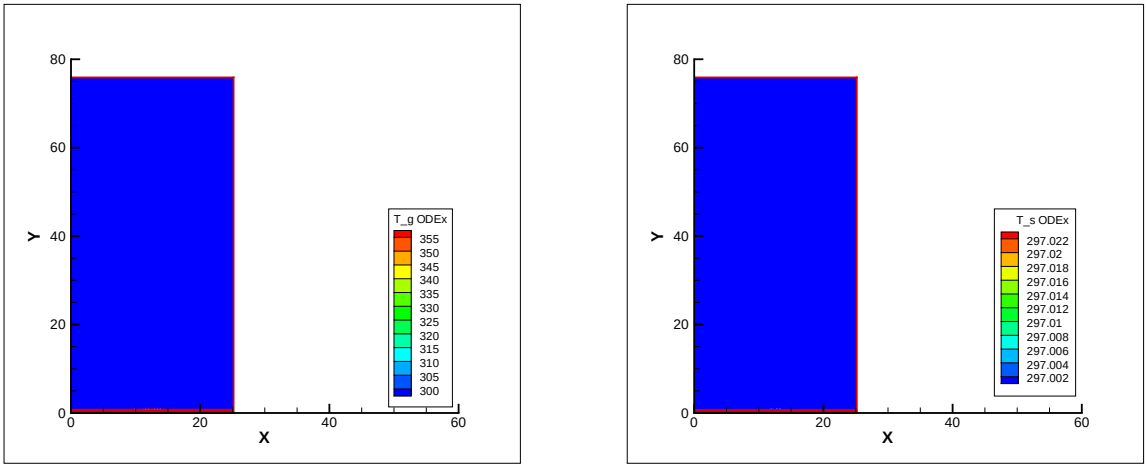


Fig. 24. Contour plots at $t_{physical} = 0.2$ sec. Showing T_g and T_s gas and solids temperature distribution using MFIX.

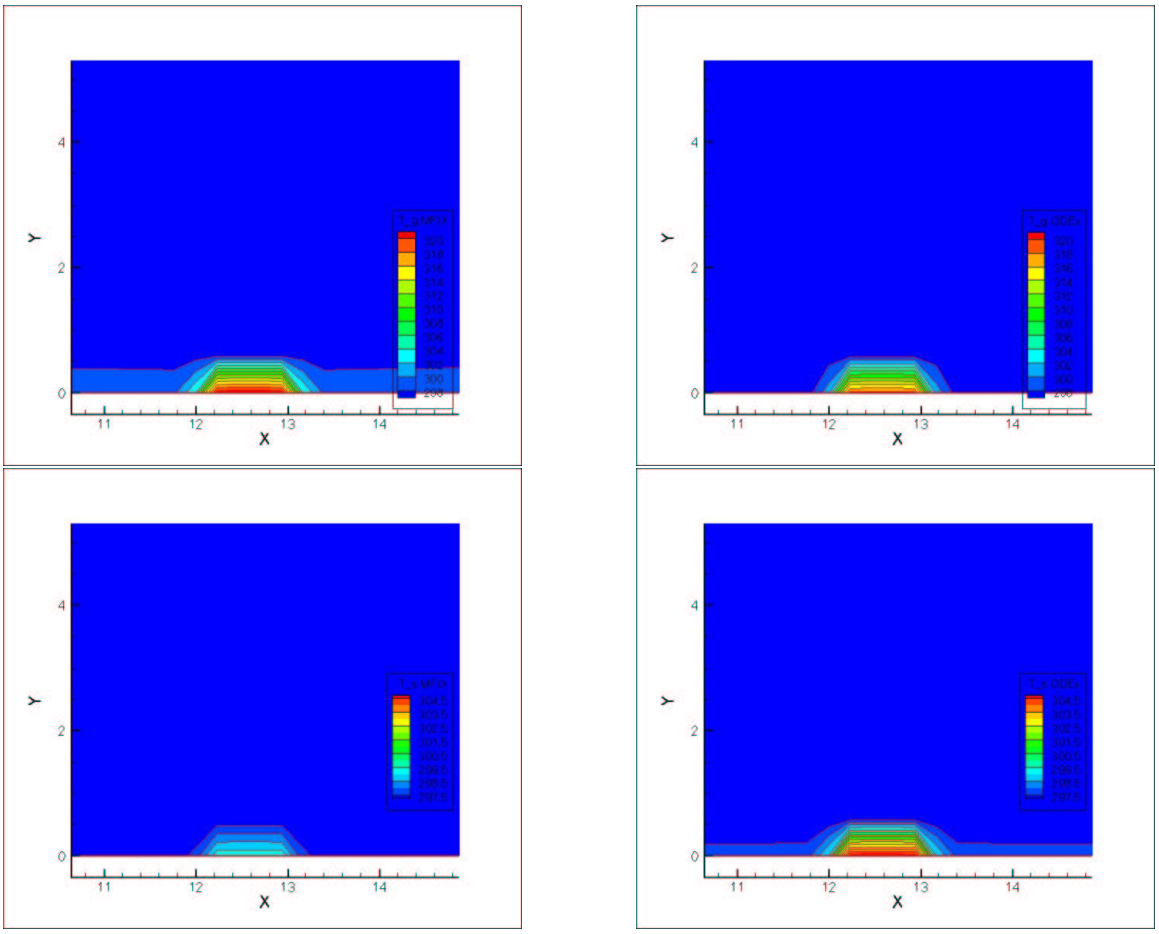


Fig. 25. Contour plots at $t_{physical} = 1.0$ sec. Showing T_g and T_s gas and solids temperature distribution using MFIX and ODEt.

In general, the figures show good agreement between the MFIX and ODEt final temperature distribution results. For this mode set, the required CPU time for ODEt is 1037.124 seconds. The calculation requires 11493.346 seconds of CPU time using MFIX. Therefore the speed-up factor for this case is approximately 11.

C. Case III: Isothermal Flow, Steady Time Extrapolation

There is much interest in the use of the POD model to predict the pattern of flow features beyond the time domain of the database. This section will present time

extrapolated results for the non-isothermal Case I conditions using ODEx. The physical parameters, geometry and boundary conditions of Case III are identical to those of Case I except that ODEx was allowed to run for an additional 0.5 seconds past the physical time domain of the snapshot database. A summary of the physical parameters, geometry and boundary conditions can be found in Table V, Figure 5(a), and Figure 5(b), respectively. For this application of time extrapolation, the field variables were computed in the time domain of the snapshot database until reaching a slowly varying state. Time extrapolation began at $t_{physical} = 1.0$ second. At this time all of the field variables had reached a slowly varying state. This case tested the ability of the POD model to predict field variable values outside of the original time domain.

For the test case chosen, the number of modes used were $Np_g = 2$, $Nu_g = 2$, $Nv_g = 5$, $Nu_s = 8$, $Nv_s = 5$, $N\epsilon_g = 5$. This mode combination differs slightly from the one used in Case I because it was discovered that this mode selection produced slightly better results for Case III. The average required CPU time for ODEx was 1123 seconds and for MFIX was 12384 seconds. As for Case I, ODEx is 11 times faster than MFIX for this case. Unlike the previous sections the time coefficients will not be qualitatively compared because the time scale is different and are therefore no longer comparable. Shown in Figures 26 and 27 are the reconstructed solutions of the POD model and the exact solutions of MFIX for the six field variables p_g, ϵ_g, u_g, v_g, u_s and v_s respectively at $t_{physical} = 1.5$ seconds. The figures show that the time extrapolation produces similar flow features to those generated in Case I. This fact is not surprising since the flow should be at a slowly varying condition at $t_{physical} = 1.0$ and 1.5 seconds.

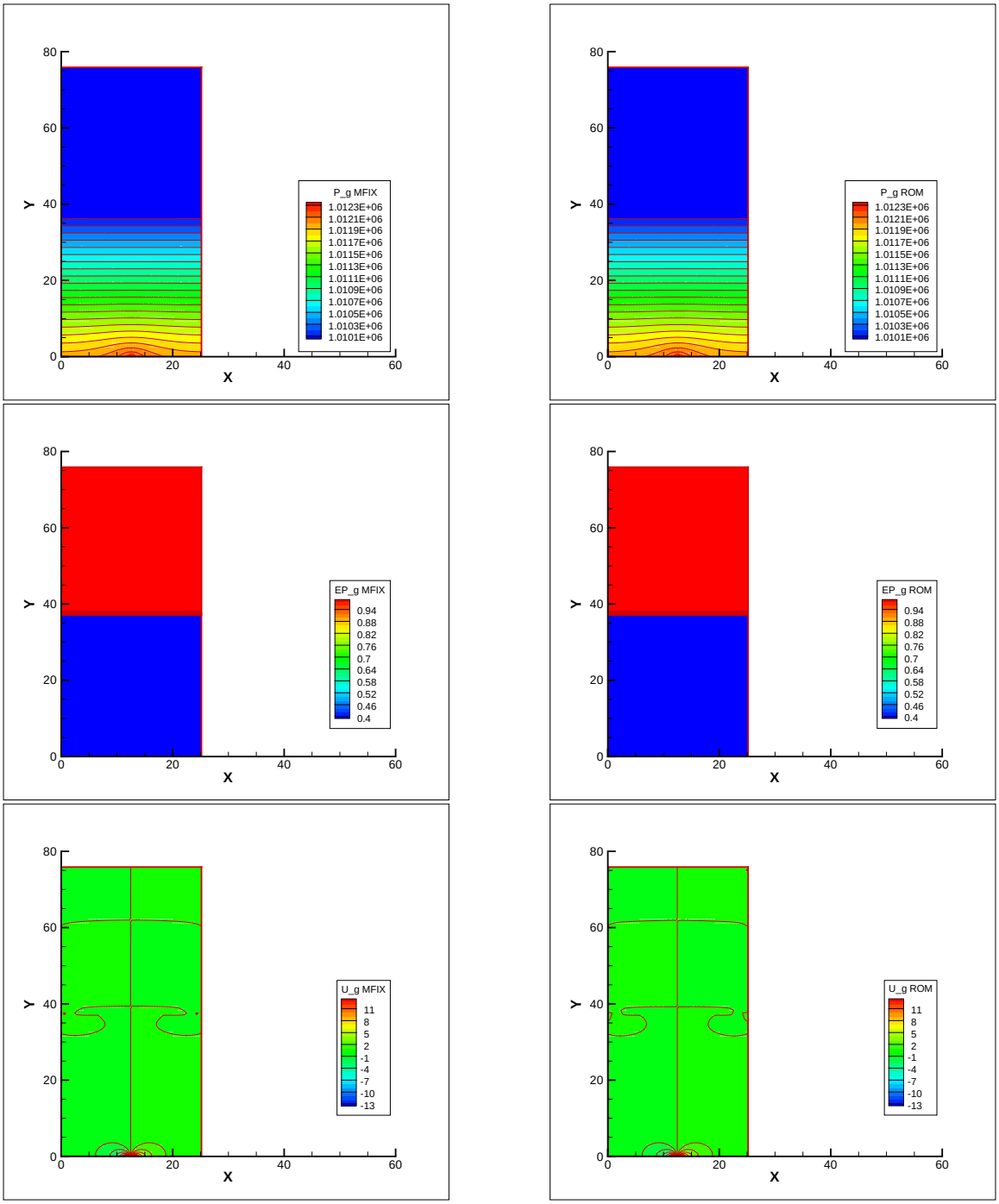


Fig. 26. Contour plots at $t_{physical} = 1.5$ sec. showing gas pressure (p_g), void fraction (ϵ_g) and x-direction gas velocity (u_g) using MFIX and ODEx time extrapolation.

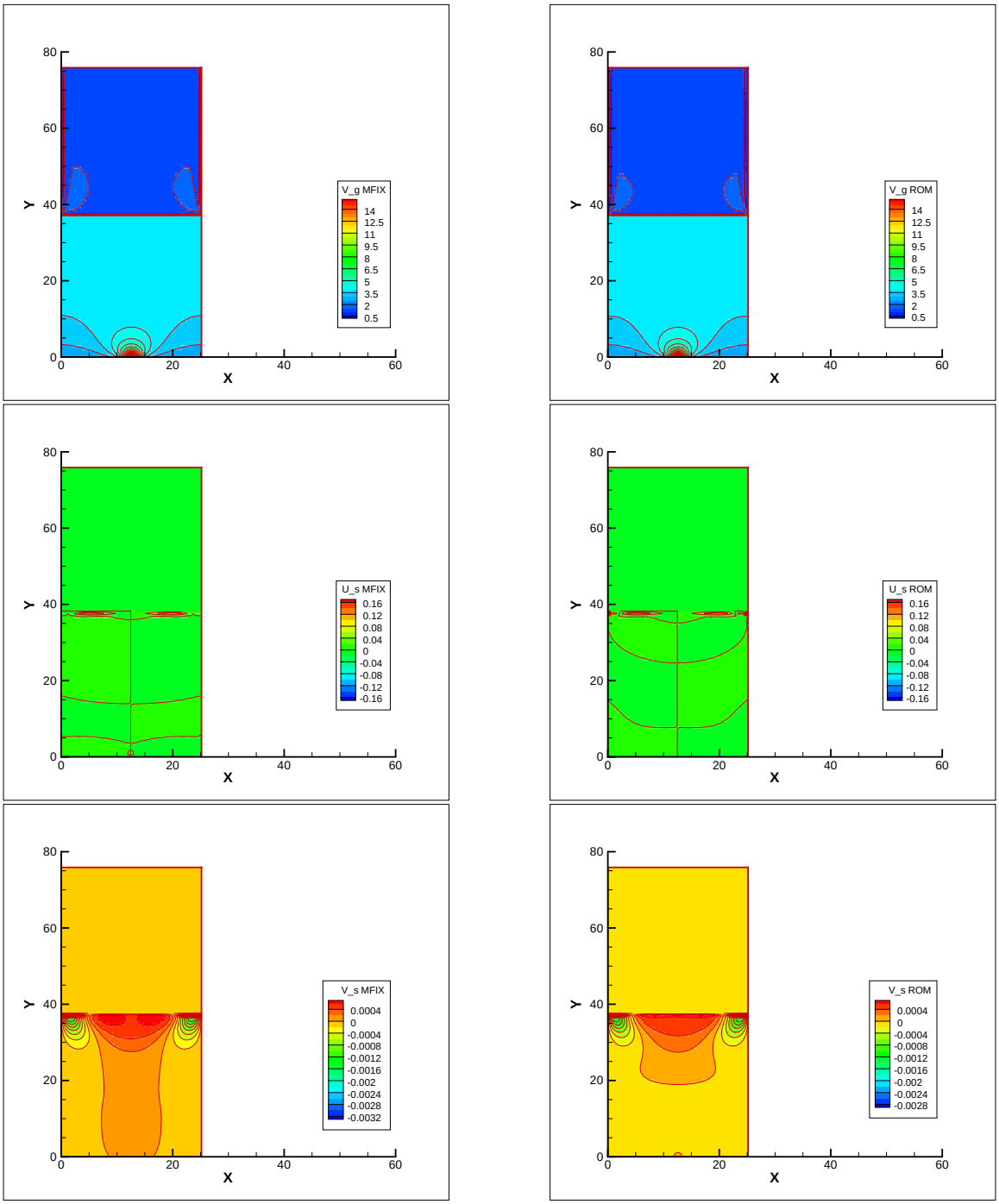


Fig. 27. Contour plots at $t_{physical} = 1.5$ sec. showing y-direction gas velocities (v_g) and the x- and y-direction solids velocities (u_s, v_s) using MFIX and ODEx time extrapolation.

The ROM did have some difficulty extrapolating the solids velocities. This seems reasonable given the difficulty of the ROM to calculate the first time coefficient of u_s in Figure 1 of Case I and the errors in the final solution of v_s in Case I.

Figure 1 of Case I shows that for the first time coefficient of u_s the time coefficients calculated using ODEx begin to deviate away from the 'exact' POD solution near the end of the time integration. It is possible to extrapolate the ROM time coefficients and plot them with the POD time coefficients for the on-reference time condition. This is shown for the time coefficients that correspond to the first mode of u_s in Figure 28. If one was to imagine a time extrapolation of the POD time coefficients in Figure 28

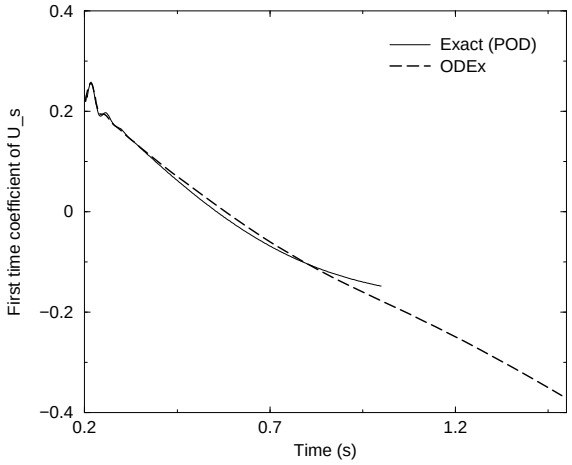


Fig. 28. Case III: The first time coefficients of u_s using time extrapolation with ODEx compared to the 'exact' on-reference solution, $t_{physical}$ = 0.2 - 1.0 seconds.

and assuming that the line continued along its path for an additional 0.5 seconds, it is obvious that the error between the Exact values and ODEx values would grow with time. Since $\alpha_1^{u_s}$ carries a significant percentage of the reconstructed solution, the result of this error appears to be that the maximum u_s velocity is much higher than that calculated by MFIX. The errant velocity only occurs at isolated locations, does not change the overall flow pattern and is not obvious in Figure 27, but it is present in the final results. Fewer modes were used to reocnstruct v_s because this produced

better results than the original mode combination, shown in Table VIII, used for Case I. Like u_s, v_s is difficult to extrapolate with good accuracy. The dominant flow features of v_s, located near the top and sides of the solids bed are well captured. However, in the middle of the bed where the v_s velocities are just above zero and either positive or negative, accuracy decreases.

D. Case IV: Isothermal Flow, Sinusoidally Varying Boundary Conditions

In this section two isothermal cases generated by MFIX are compared. Within this section one case will be referred to the steady case and the other will be referred to as the unsteady case. In steady case gas is injected with a steady velocity as done in Case I. In the unsteady case gas is injected sinusoidally as a function of time. In

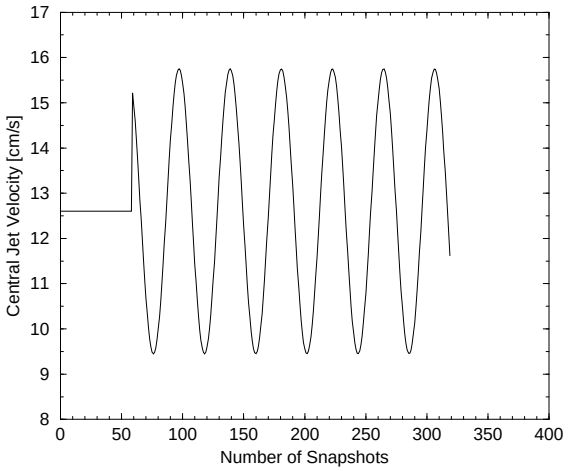


Fig. 29. Variation of the central jet at the boundary.

both cases the geometry and grid size were the same and are shown in Figure 5. The physical parameters of both cases are shown in Table V with the exception that for the unsteady case v_1 varies with time. In the unsteady case, gas is injected through the lower boundary at 12.6 cm/s until the end of the transient phase which occurs during the interval 0.2 to 0.35 seconds of physical time. At $t_{physical} = 0.35$

seconds the boundary condition of the central jet was changed to inject gas through the lower boundary with velocity $12.6 + 3.15\sin(60t)$ cm/s where t is the current physical time. the maximum Figure 29 shows the variation of the central jet velocity. The MFIX results were used to generate 320 snapshots from 0.2 to 1.0 seconds. The time coefficients and basis functions were extracted by applying PODDEC to the snapshots generated by MFIX. The following ten plots in Figures 30 and 31 show the cumulative energy spectra of the modes for gas pressure, $u-$ and $v-$gas velocities, and $u-$ and $v-$solids velocities (p_g, u_g, v_g, u_s and v_s) for the steady jet and varying jet cases respectively.

Another method to analyze the affect of the number of modes on the solution is to look at the value of the reconstruction of each mode relative to the average mode over the whole time domain. Recall that the reconstruction of any variable decomposed into time independent basis functions (ϕ) and time dependent time coefficients (α) is given by

$$\mathbf{u}(\mathbf{x}, t) \simeq \phi_0(\mathbf{x}) + \sum_{j=1} \alpha_j(t)\phi_j(\mathbf{x}) \tag{5.1}$$

where ϕ_0 denotes the average mode of the variable. Calculating the spatially averaged percentage of each $\alpha_j * \phi_j$ product compared to the average ϕ_0 mode for each j quantifies affect of each mode on the reconstructed solution. More precisely this quantity, labeled $P_j(t)$, is given by:

$$P_j(t) = 100 * avg_{ij}\left(\frac{\sum_{ij=1}^{n} \alpha_j(t)\phi_j(ij)}{\phi_0(ij)}\right) \tag{5.2}$$

where n is the total number of spatial points and avg_{ij} indicates the spatial average over the points. Figure 32 shows $P_j(t)$ for the steady and unsteady cases.

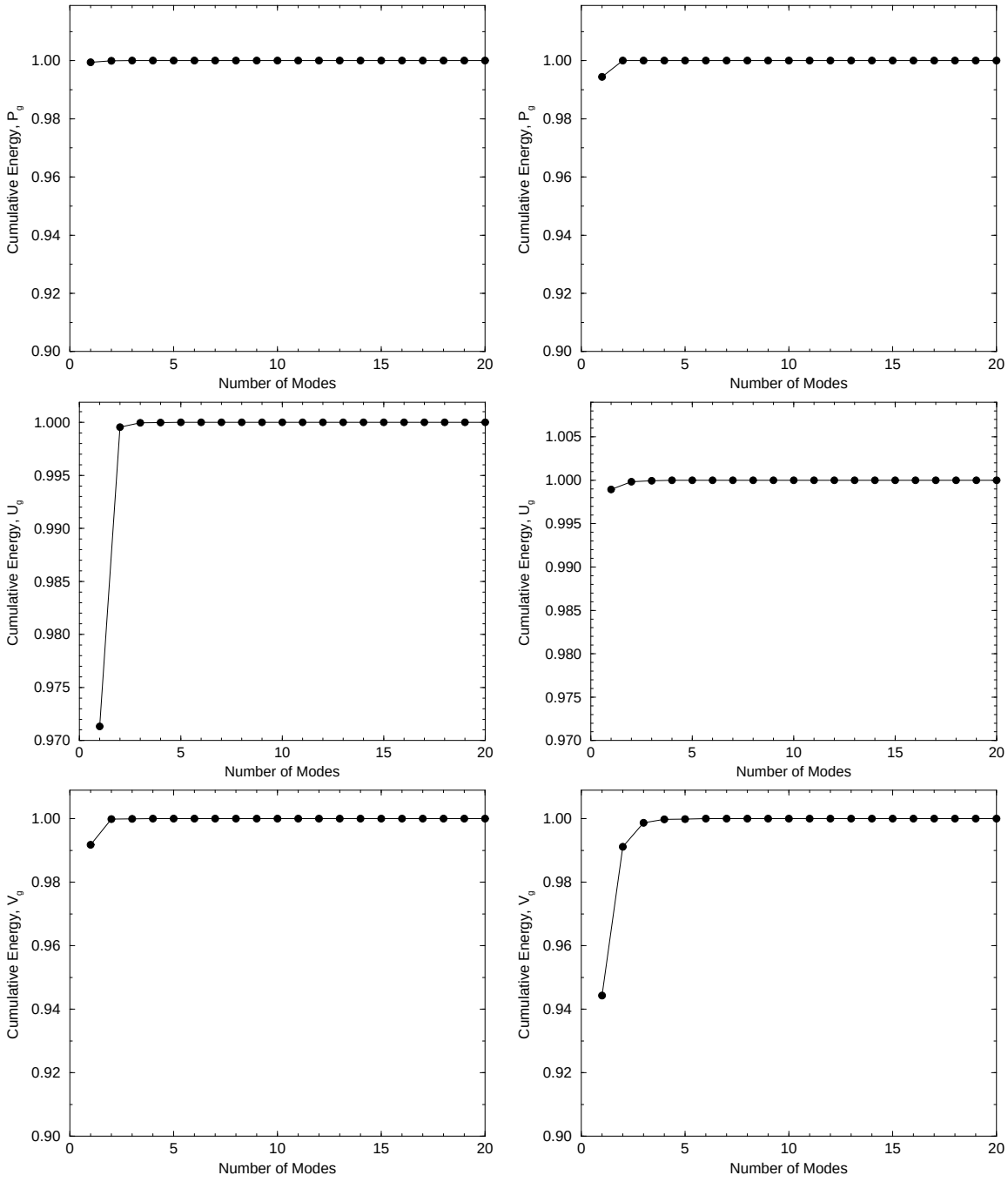


Fig. 30. Cumulative energy spectra for p_g, u_g and v_g for the steady (left) and varying (right) jet cases.

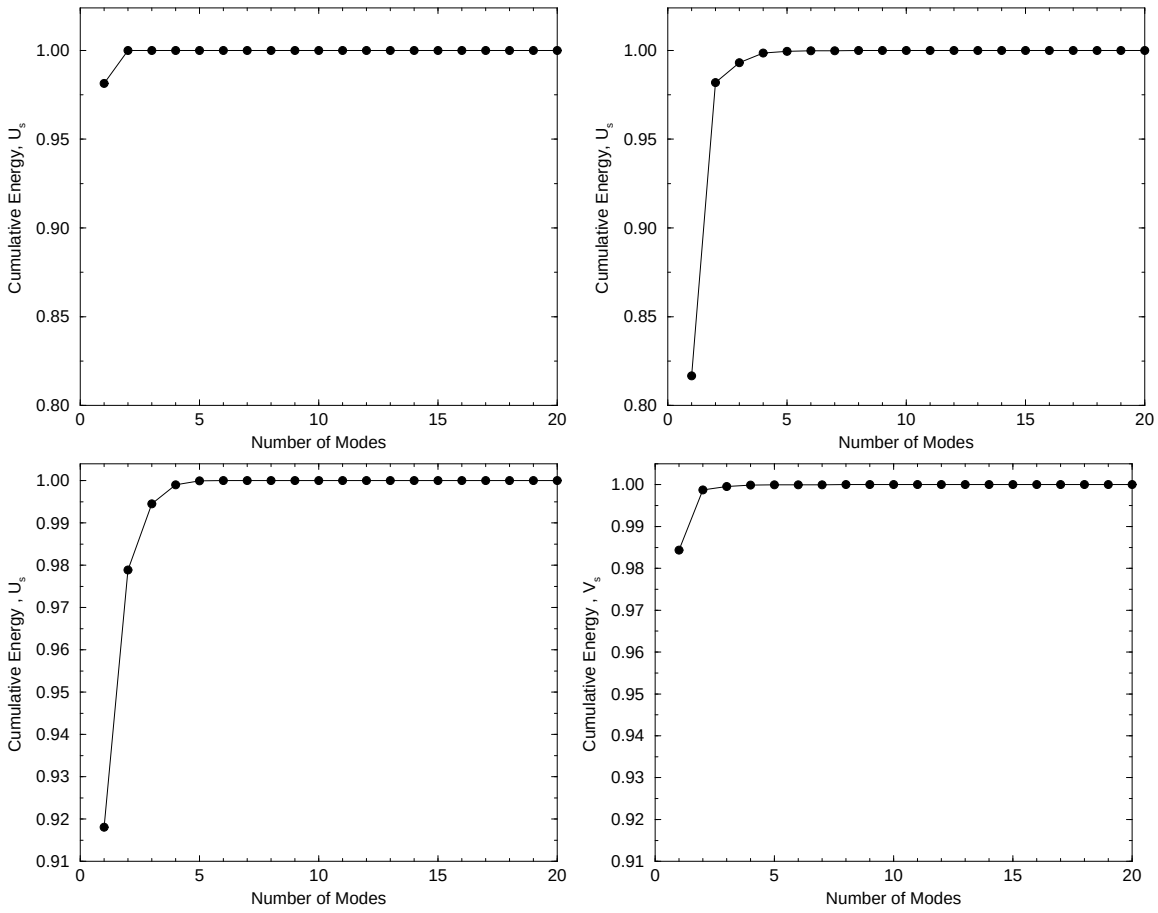


Fig. 31. Cumulative energy spectra for u_s and v_s and time history of the central jet at the boundary for the steady (left) and varying (right) jet case.

The plots show values for gas pressure, u- and v-solids velocities respectively. All of the percentage values are absolute values. $P_j(t)$ gives the relative weight of each reconstructed mode when compared to the average mode with respect to time. For example, consider the last two plots in Figure 32 that show the weight of the reconstructed mode values of v_g. For the steady jet case, the reconstructed first mode only represents a maximum of approximately 1% of the value of the average mode value. This means that the first mode would only contribute a very small amount to the reconstructed solution. By contrast, the weight of the first reconstructed mode value of v_g for the varying jet case contributes a maximum of 7% of the average. Furthermore, the oscillations shown in the plot indicate that the time coefficient corresponding to this mode captures some of the sinusoidal variation of the velocity which is not present in the average mode. The plots in Figure 32 show that, in general, the reconstructed modes of the varying velocity case contribute to a larger part of the solution than the modes of the steady jet case. Figures 33 - 36 show contour plots of the modes extracted by PODDEC for each case. The plots show the first several modes for gas pressure $u-$ and v-gas velocities respectively. The steady jet case is shown on the left and the varying jet case is shown on the right. The average mode for each variable is labeled 'Mode 0'. The figures show that the maximum and minimum values are usually larger for the varying jet case than the steady case and that the varying jet modes contain more features than the steady jet case. The contour plots show that the average mode of v_g is almost identical for the steady and varying cases.

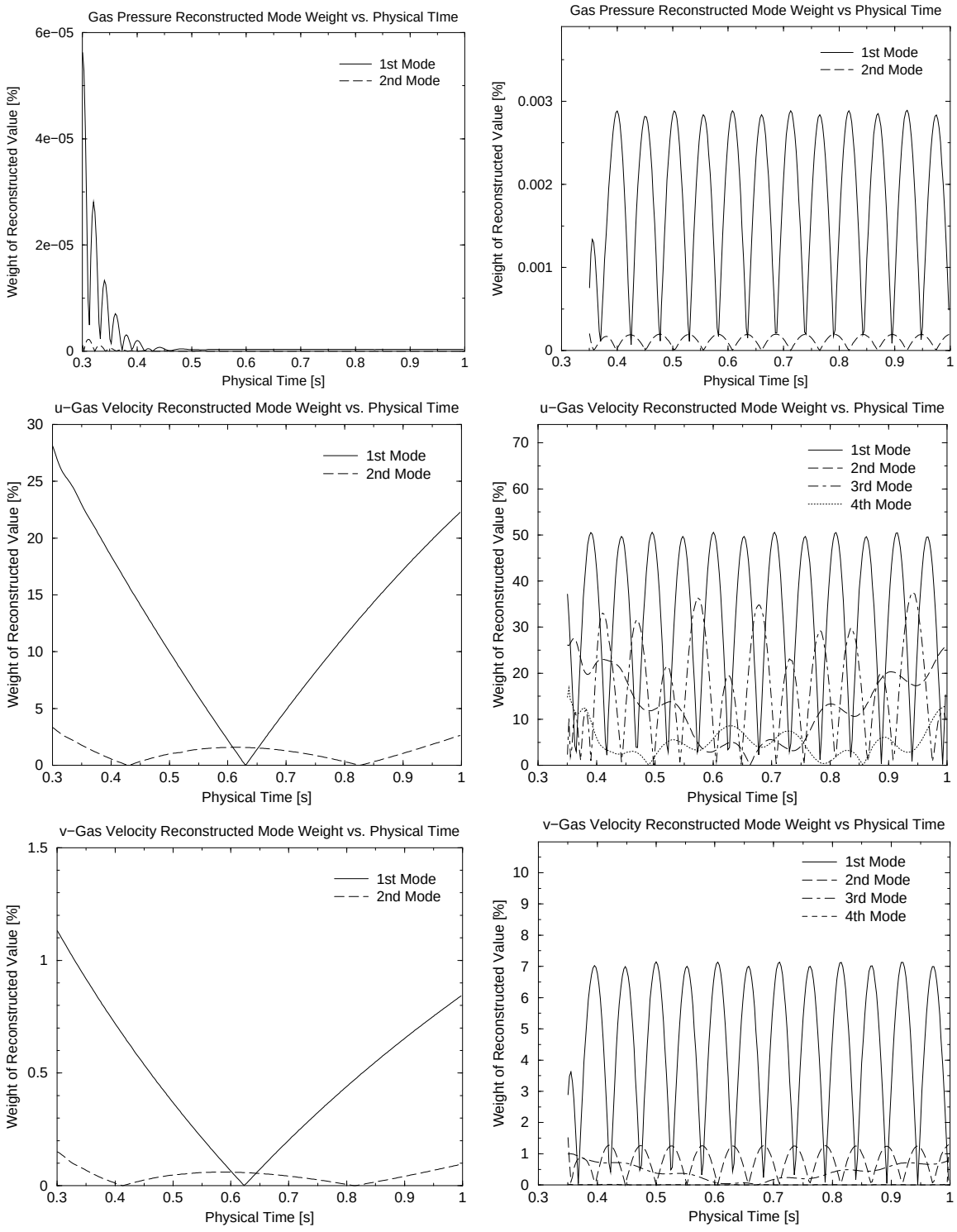


Fig. 32. Time history of the weight of each reconstructed mode value compared to the average mode averaged over the spatial domain for the steady (left) and varying (right) jet cases.

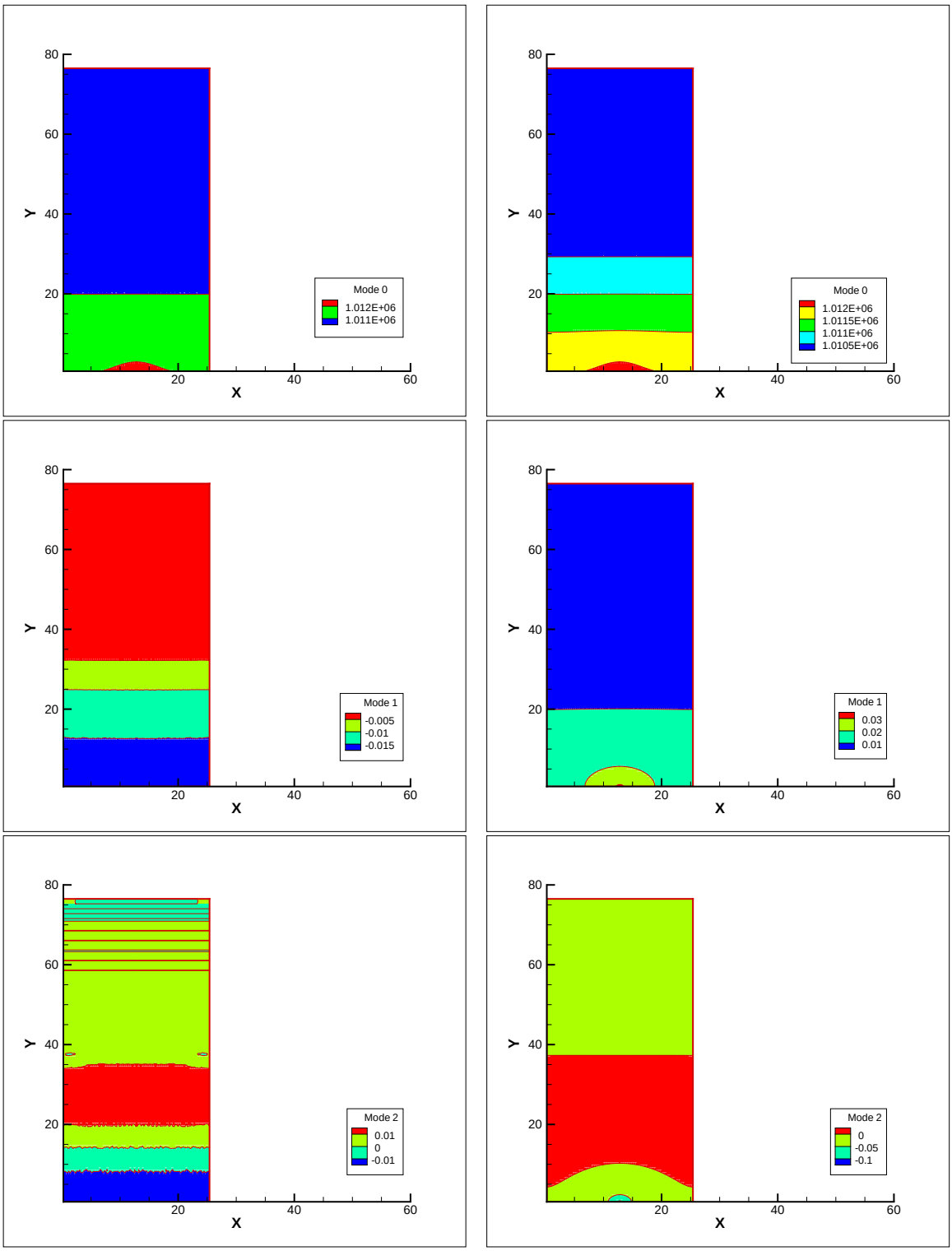


Fig. 33. The average and first two modes of gas pressure, p_g, for the steady(left) and varying(right) jet cases.

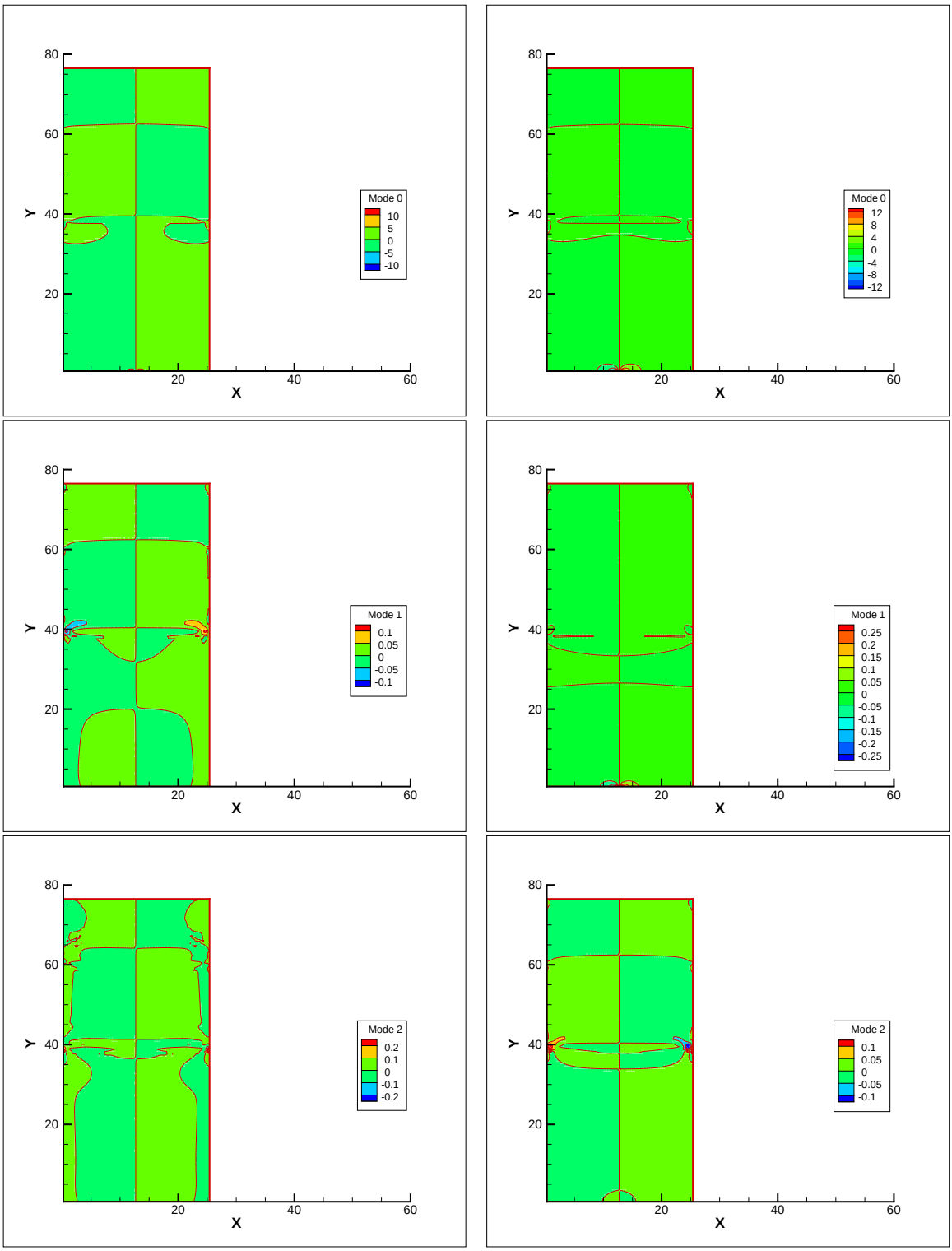


Fig. 34. The average and first two modes of u_g velocity for the steady(left) and varying(right) jet cases.

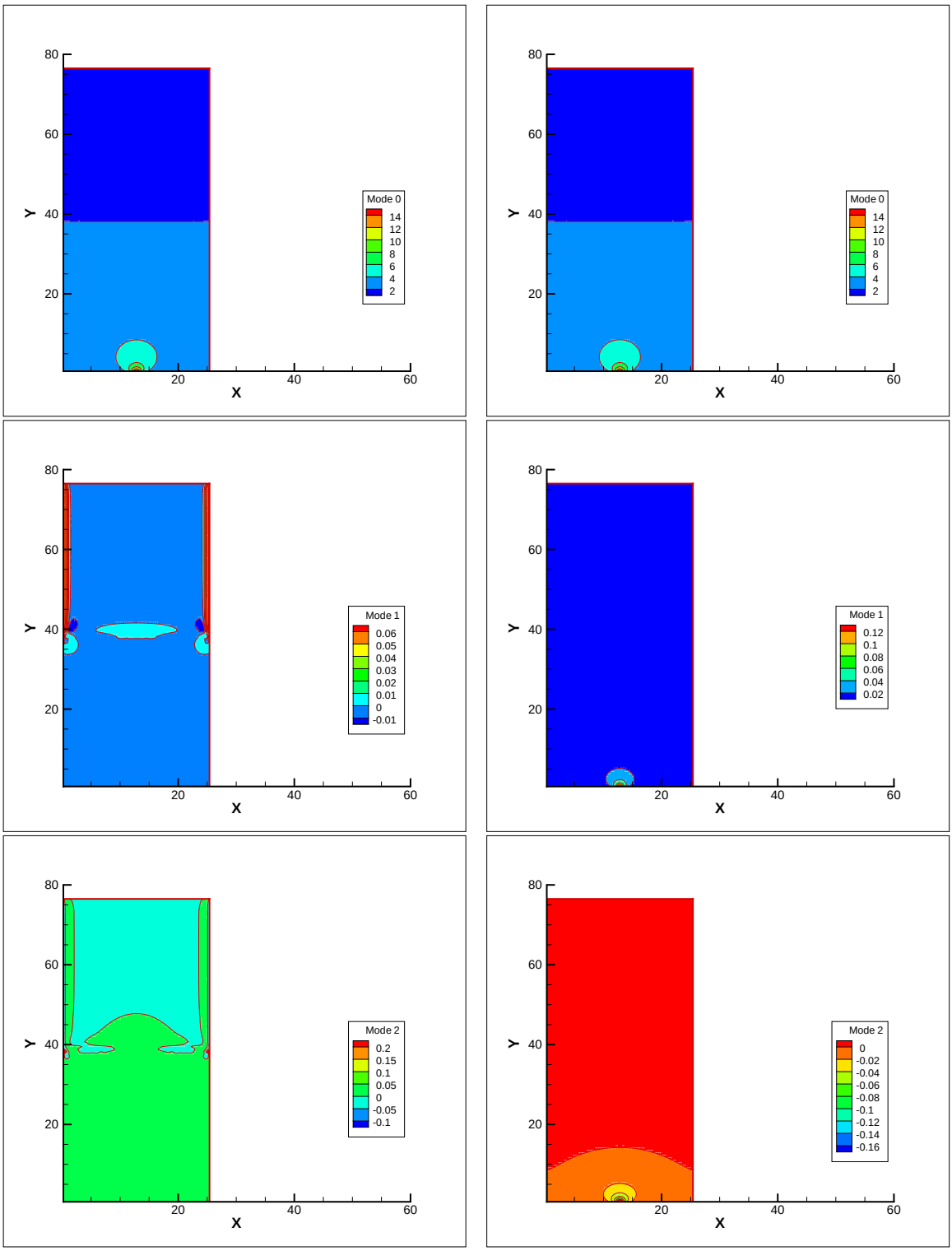


Fig. 35. The average and first two modes of v_g velocity for the steady(left) and varying(right) jet cases.

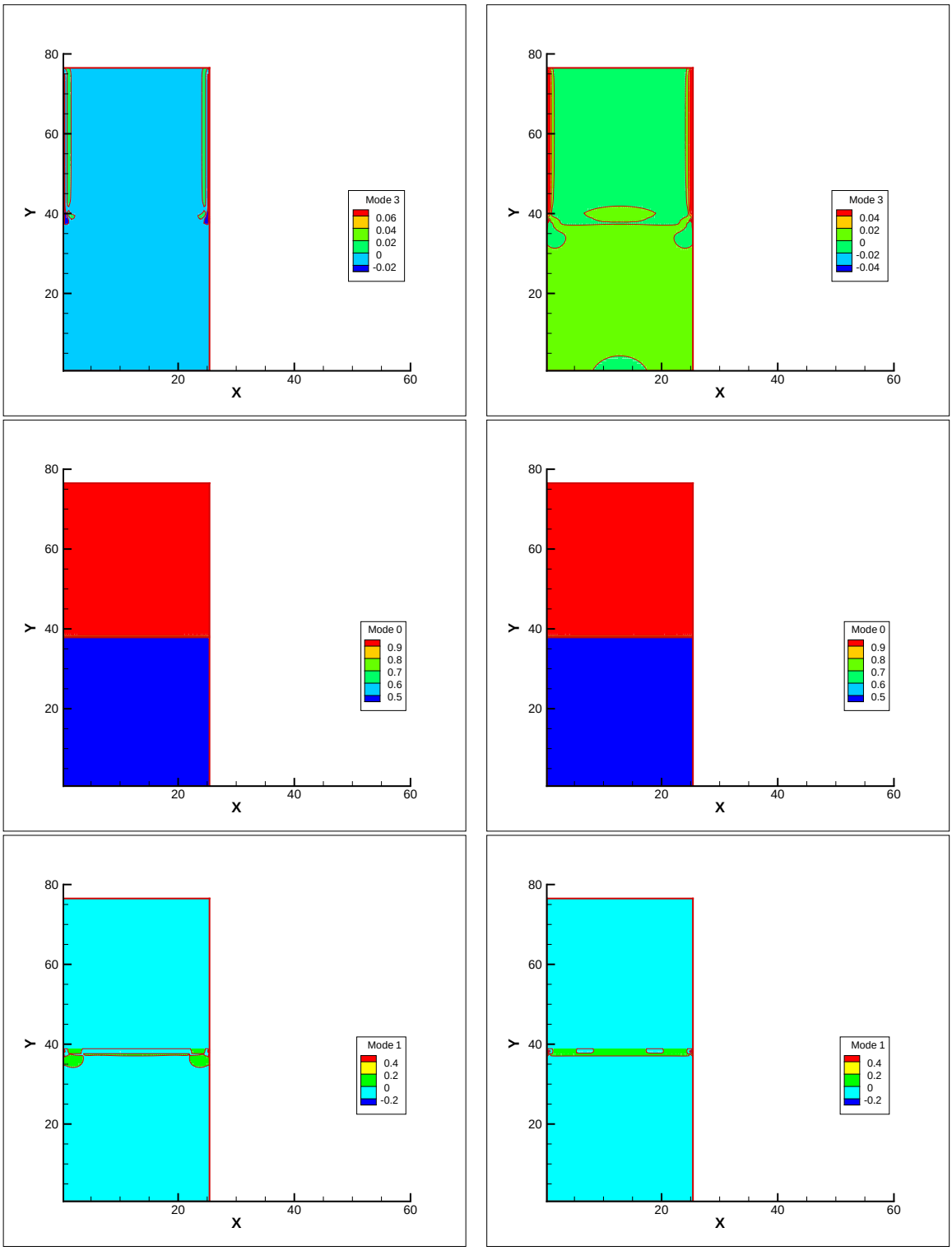


Fig. 36. The fourth mode of v_g velocity and the average and first mode of void fraction, ϵ_g, for the steady(left) and varying(right) jet case.

Results for the unsteady injection velocity case using ODEx will now be presented in this section. Simulation of this periodic injection velocity case required time dependent boundary conditions to be specified at the inlet for the ODEx algorithm. The magnitude of the velocity of the central gas jet inlet was varied in ODEx by $12.6 + 3.15\sin(60t)$ cm/s. This boundary condition was specified because the POD basis modes do not communicate to ODEx the time variation specified by the original boundary condition used in the full-order model to generate the database.

The mode set chosen for this simulation is shown in Table XI under the column labeled, Case A. The required CPU time for ODEx is 472 seconds. The required CPU time for MFIX is 10170 seconds. Therefore, this mode combination resulted in a speed-up factor of 21.5. Figures 37 and 38 show plots of the 'exact' time coefficients extracted from the database using PODDEC and those computed by ODEx. These figures show that good correlation exists between the POD and ODEx time coefficients corresponding to the first mode. Errors become larger for time coefficients corresponding to additional modes.

Table XI. Number of modes used for varying jet velocity cases.

Field variable	Symbol	Number of modes
Gas pressure	N_{p_g}	1
Void fraction	$N_{\epsilon g}$	2
u gas velocity	N_{ug}	3
v gas velocity	N_{vg}	2
u solids velocity	N_{us}	3
v solids velocity	N_{vs}	3

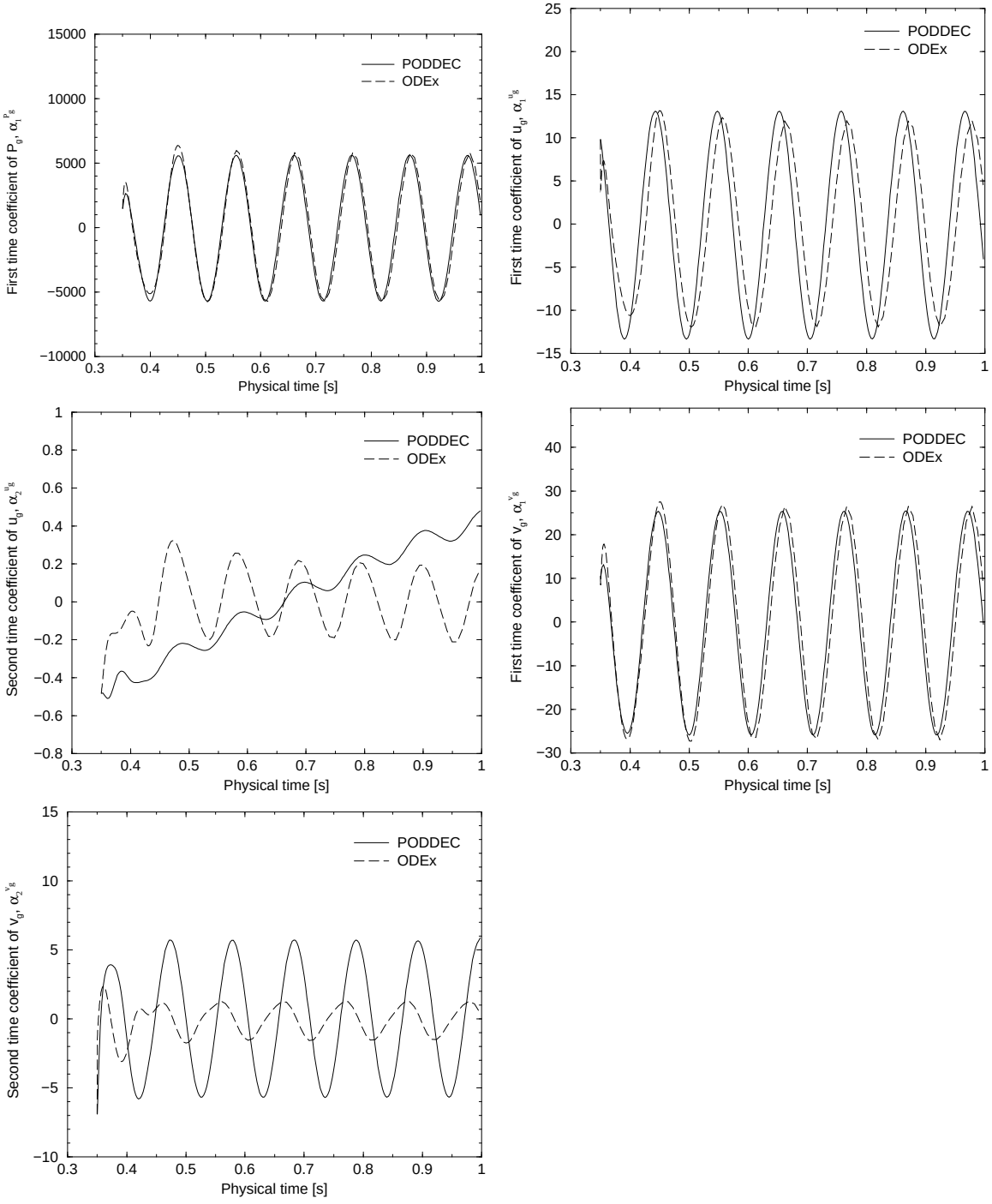


Fig. 37. Time coefficients of gas pressure $u-$ and $v-$gas velocities computed by ODEx and corresponding exact values computed by PODDEC.

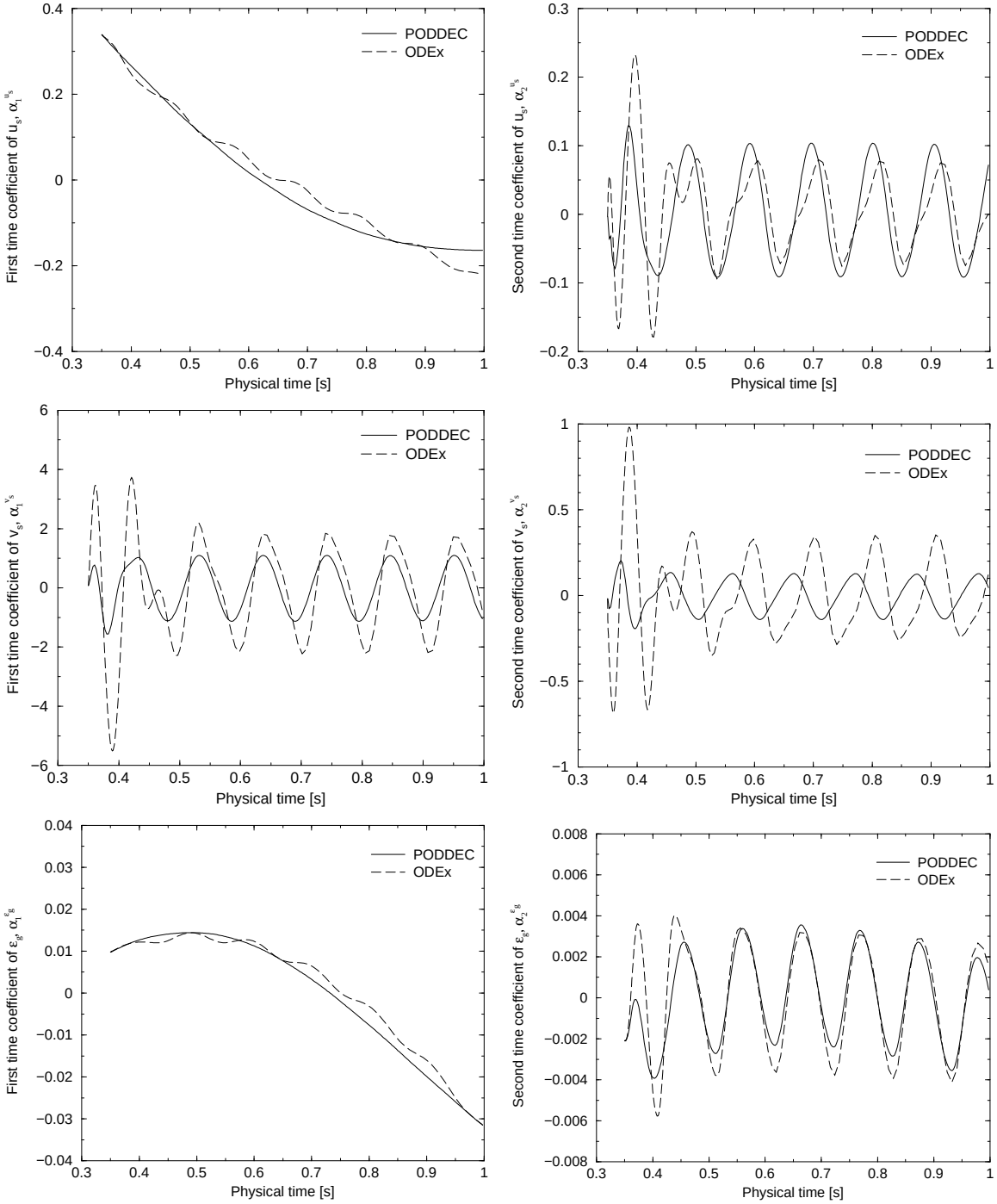


Fig. 38. Time coefficients of void fraction, $u-$ and $v-$solids velocities computed by ODEx and corresponding exact values computed by PODDEC.

E. Summary

In this chapter the results of four investigated cases were presented. The first case was an isothermal flow simulation. Investigation showed that the POD-based ROM, ODEx, produced qualitatively accurate final results and was 11 times faster than the full-order model, MFIX. The second case was similar to the first except non-isothermal flow was modeled. This investigation showed that ODEt was also 11 times faster than MFIX for the case studied. The third case was an investigation into the ability to perform time extrapolation with the POD-based ROM, ODEx. Time extrapolation generated a slight increase in the qualitative error but did not significantly decrease the required computation time. The fourth case is an investigation of the ability of ODEx to simulate a periodic time dependent boundary condition. ODEx produced good results for the first modes of the field variables. In general, errors in the time coefficients increased for additional modes in Case IV. For the periodic boundary condition case, the POD-based ROM, ODEx ran 21.5 times faster than the full-order model, MFIX. In the next chapter, the first case will be further isolated and scrutinized. The next chapter presents POD-based ROM acceleration techniques using Case I as a basis.

CHAPTER VI

ACCELERATION METHODS

The objective of this chapter is to present a set of acceleration techniques for POD-based reduced-order models. These techniques include: (i) an algorithm for splitting the snapshot database, (ii) a method for solving quasi-symmetrical matrices, and (iii) an algorithm for time step adjustment. A detailed comparison of the full and reduced-order model algorithms is described. The acceleration techniques proposed for the solution of the reduced-order model are subsequently presented.

A. Structure of Numerical Algorithms

The purpose of this section is to compare and contrast the full-order model implemented in the MFIX code with the reduced-order model implemented in the ODEx code. Differences in computation time, sub-iteration time and number of sub-iterations for a particular case are presented.

The solution algorithms used in ODEx and MFIX are similar in organization. For a two-dimensional isothermal case, both codes must solve for six dependent field variables: void fraction, gas pressure, and gas and solids phase velocities. Unlike MFIX, ODEx solves for the time coefficients of each dependent field variable and reconstructs the values from the basis functions. MFIX and ODEx use fully implicit, time marching algorithms. At each time step sub-iterations are performed until a residual criterion has been satisfied. The time step size is adjusted based on the convergence rate during the calculation.

Figure 39 shows a flow chart of the code algorithm shared by ODEx and MFIX, and the groups of common subroutines.

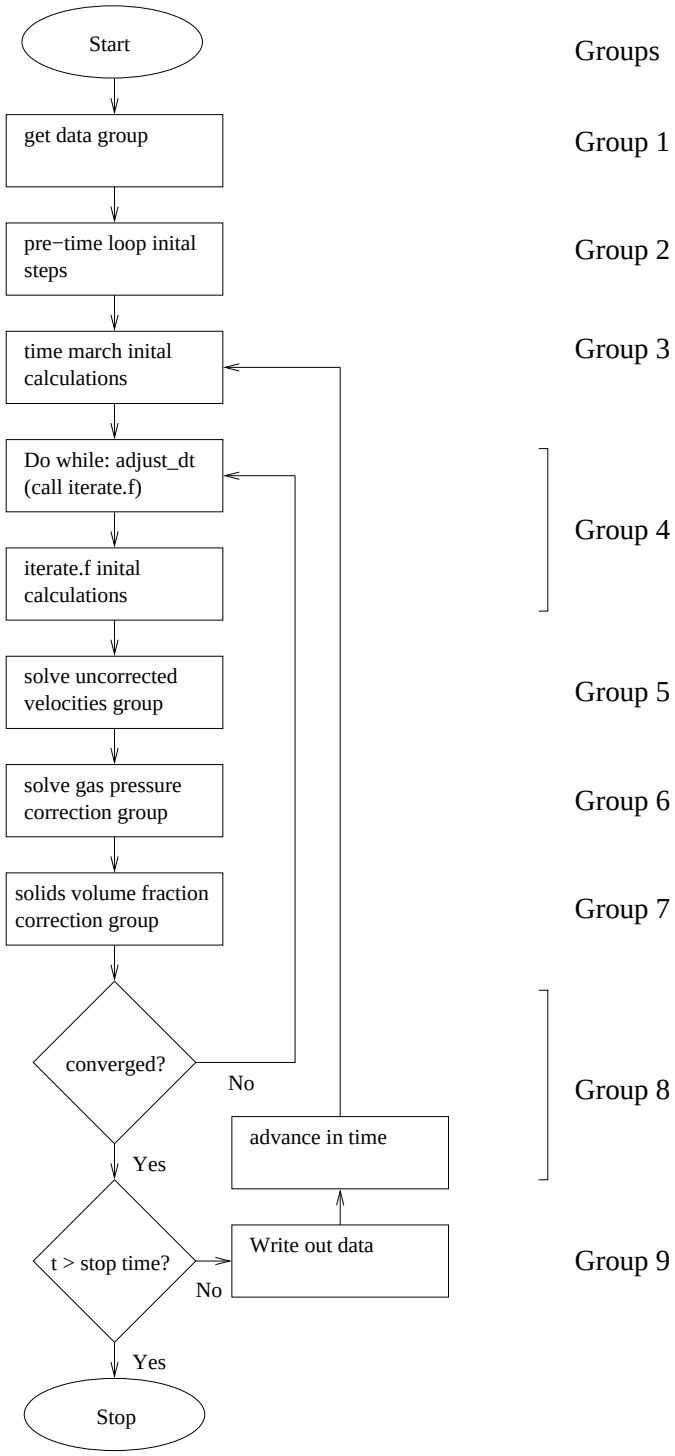


Fig. 39. General MFIX/ODEx logic flowchart.

Figure 40 shows which groups were modified for the reduced-order model and to what degree they were modified.

A comparison of the full-order and reduced-order models is presented to explain the differences in computational time per subiteration. To facilitate this comparison, the subroutines of MFIX and ODEx were divided into similar groups. Although the role of each group is the same in MFIX and ODEx, the details of the subroutines in the groups differ. The definitions of the groups are presented in Table XII.

Table XII. MFIX/ODEx group descriptions.

Group	Description
1	Reads initial data, sets boundary and initial conditions
2	Calculates and sets initial dependent variables
3	Calculates initial values inside the time loop
4	Calculates initial values inside sub-iteration loop
5	Calculates velocity values
6	Calculates pressure and corrects velocities
7	Calculates solids volume fraction and corrects velocities
8	Checks convergence and performs final steps in iteration loop
9	Writes output

Both models spent most of computational time in groups 5, 6, and 7, as shown in Table XIII. These three groups form the subiteration loop. Each of these three groups are used once during a single subiteration.

Group 7 was chosen to be explored in detail for two reasons. First, the calculations in this group are a significant percentage of the total computation time. The

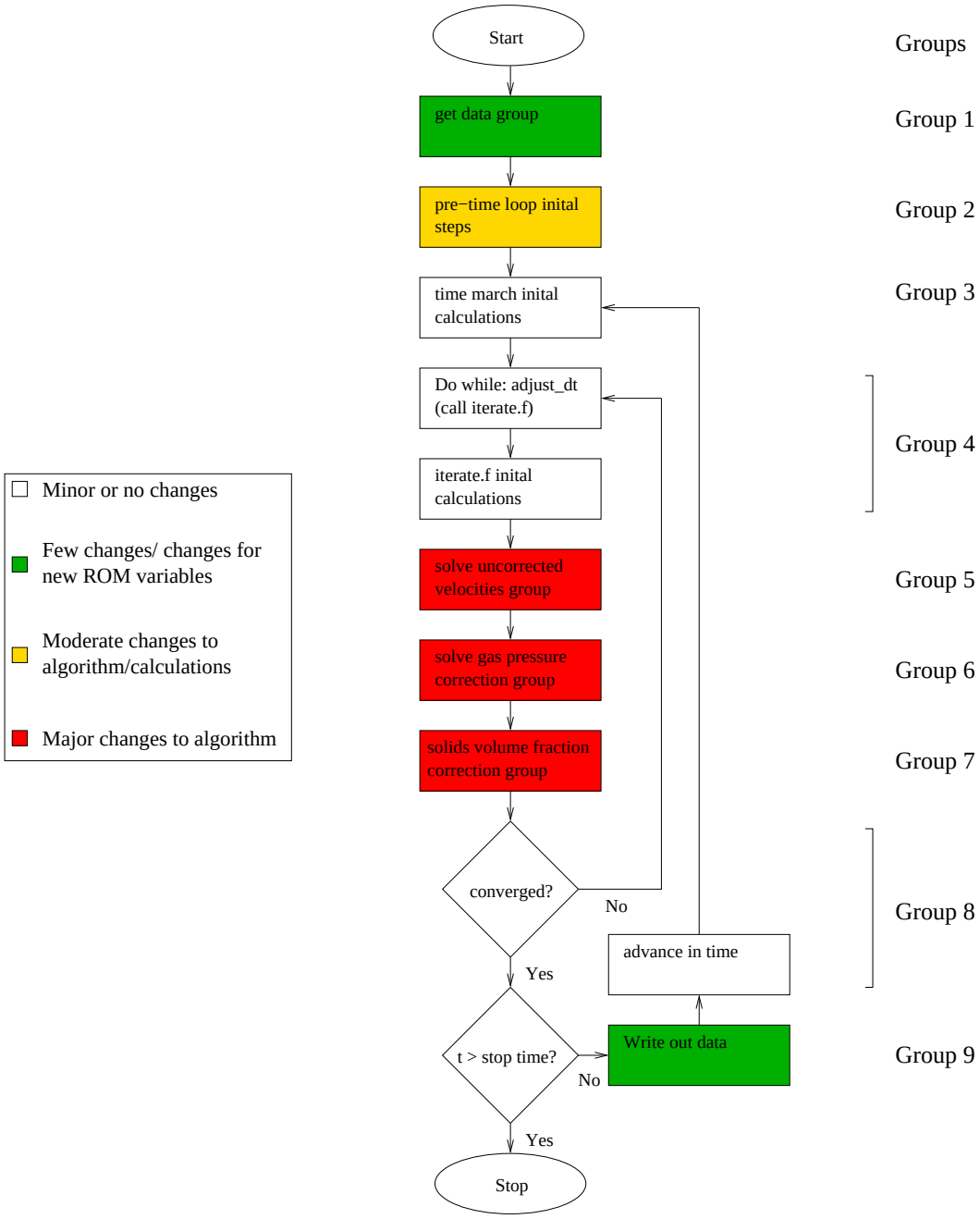


Fig. 40. Groups of subroutines modified to create ODEx.

Table XIII. Time profile of MFIX and ODEx codes.

Group	MFIX		ODEx	
	[s]	[%]	[s]	[%]
1	0.396	0.004	0.874	0.097
2	0.586	0.006	0.007	0.001
3	171.655	1.794	4.987	0.555
4	0.231	0.002	25.727	2.863
5	5794.622	60.574	563.552	62.729
6	2331.241	24.370	82.377	9.169
7	1162.497	12.152	213.978	23.817
8	83.266	0.870	6.256	0.696
9	21.670	0.227	0.644	0.073
Total	9566.164	100.00	898.402	100.00

calculations in group 7 are approximately 24% of the total computation time of ODEx and 12% for MFIX. Second, numerical testing showed that to achieve a good solution, the void fraction often required more modes than the other field variables. Figure 41 lists, for comparison, the major subroutines of group 7 in MFIX and ODEx.

The major differences between group 7 of ODEx and MFIX are the addition of the calc_ab_pp_s, reconstruct_u_s and reconstruct_v_s routines in ODEx, the conglomeration of calc_vol_frac.f into correct_1.f and the removal of adjust_leq. The subroutine calc_ab_pp_s performs the projection of the solids volume fraction correction equation onto the basis functions described in equation (4.18). The reconstruct routines regenerates the field variables u_s and v_s from their corrected time coefficients. The subroutine adjust_leq was used by MFIX to adjust the type of solver used to solve

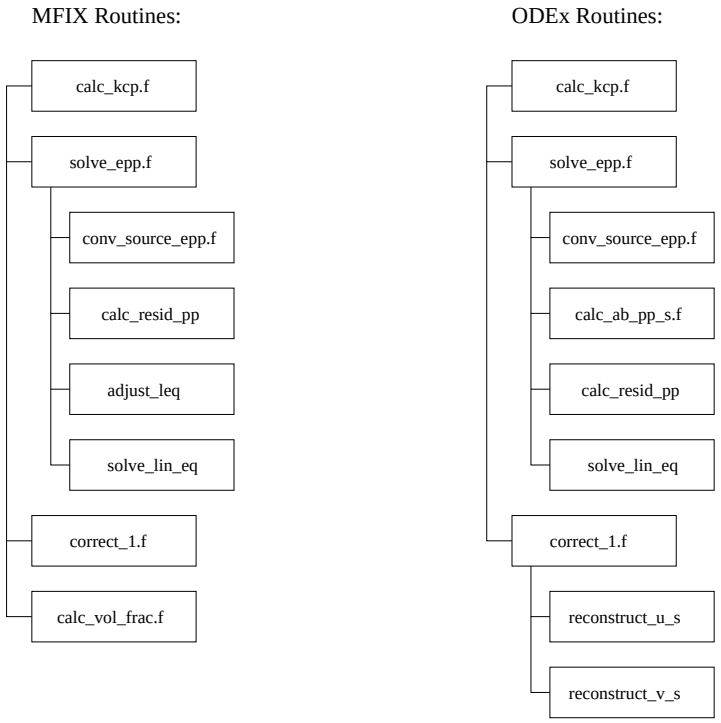


Fig. 41. Structure of the code for group 7.

the system of equations. This routine is not necessary in ODEx because the system of ODEs are always solved using the same solver.

To determine the most computationally expensive calculations, a time profile of group 7 was generated for both MFIX and ODEx. These time profiles are shown in Fig. 42. Figure 42(a) reveals that the time profile of MFIX was dominated by the subroutine that solves the linear set of equations (3.1) for the solids volume fraction at each grid point. In contrast, the profile of ODEx was dominated by the projection of the basis functions. In ODEx, the computational time spent for the solution of the linear systems of equations was less than 1% of the total computational time. If less than three modes are used, the reconstruction of the field variables, which is part of the correction group, uses the majority of the calculation time. The conclusions drawn based on the solids volume fraction hold for all the dependent field variables.

To further determine the difference between the reduced-order model and the full

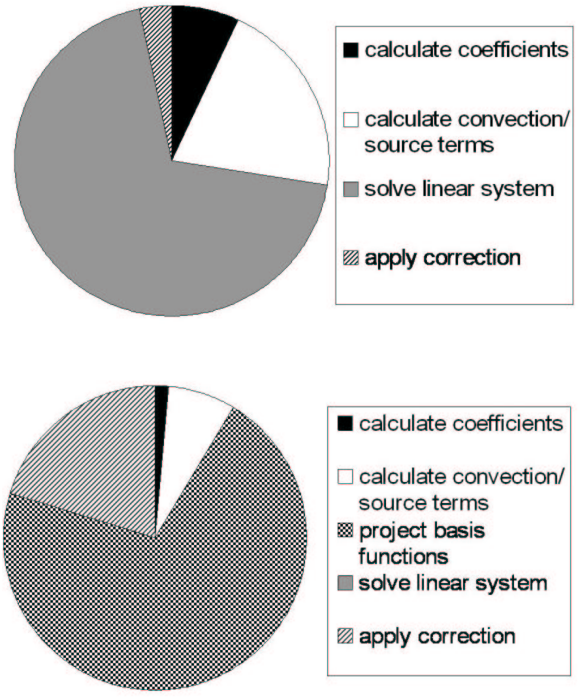


Fig. 42. Time profile of group 7: a) MFIX, and b) ODEx.

model, the number of operations was counted in the projection subroutines of ODEx.

Table XIV shows that the number of operations in ODEx varies as a quadratic

Table XIV. Number of operations estimates for the projection routines in ODEx.

Variable	Additions	Multiplications
u_m	$(IMAX2-3)(JMAX2)(8{N_{um}}^2+9N_{um})$	$(IMAX2-3)(JMAX2)(6{N_{um}}^2+7N_{um})$
v_m	$(IMAX2-2)(JMAX2)(8{N_{vm}}^2+9N_{vm})$	$(IMAX2-2)(JMAX2)(6{N_{vm}}^2+7N_{vm})$
p_g	$(IMAX2-2)(JMAX2-2)(6{N_{pg}}^2+2N_{pg})$	$(IMAX2-2)(JMAX2-2)(7{N_{pg}}^2+2N_{pg})$
ϵ_g	$(IMAX2-2)(JMAX2-2)(6{N_{\epsilon g}}^2+2N_{\epsilon g})$	$(IMAX2-2)(JMAX2-2)(7{N_{\epsilon g}}^2+2N_{\epsilon g})$

Table XV. Example number of multiplications performed in the routines that dominate the CPU time of MFIX and ODEx.

Code	IMAX	JMAX	N_{um}	N_{vm}	$N_{pg}/N_{\epsilon g}$	Multiplications
ODEx	108	124	2	2	2	2.93E6
ODEx	108	124	4	4	4	9.93E6
MFIX	108	124	-	-	-	1.16E6
ODEx	54	62	2	2	2	0.73E6
ODEx	54	62	4	4	4	2.50E6
MFIX	54	62	-	-	-	0.30E6

function of the number of modes. In MFIX, the number of total operations was approximately $28(IMAX+2)(JMAX+2)$.[24] Therefore the number of operations per subiteration in ODEx is always greater in ODEx than in MFIX[1]. In spite of this,

[1]Except for the atypical case when only one mode is used to represent each field variable

numerical tests showed that using the mode combination in Table VIII in the Results section, ODEx produced accurate results and was 11 times faster than MFIX. The speed-up is due to the fact that ODEx required fewer subiterations per time step than MFIX. This is the result of the fact that the time step limitations in the reduced-order model that solves ODEs were less restrictive that those of the full-order model that solves PDEs. Consequently, ODEx could use larger time steps than MFIX.

B. Acceleration Methods

This section describes several methods developed to further decrease the computational time of the reduced-order model. Four techniques are presented herein: (i) an algorithm for splitting the database, (ii) an algorithm for solving quasi-symmetrical matrices, (iii) a strategy for reducing the frequency of updating $\tilde{\mathcal{A}}$, and (iv) a time step adjustment method.

1. Database Splitting

The POD basis functions are extracted from a database of snapshots generated by numerically integrating the governing differential equations. Currently, it is common to use a database that includes all the snapshots. Using a single database that covers the entire time domain, however, could be too restrictive. For example, consider the transience during the startup of the flow in a fluidized bed. The large time variation at startup requires more modes than are necessary to model the flow features present in the latter part of the simulation. A method to avoid this problem is to split the database of snapshots.

Splitting the database into multiple subsets produces an auto-correlation matrix $\overline{\overline{R}}$ that contains more relative energy in the first modes. Herein, energy is defined

as the sum of all the POD eigenvalues. The relative energy captured by the kth mode is defined as $\lambda_k / \sum_{j=1}^{M} \lambda_j$.[21] As the relative energy of the first modes increases, fewer POD modes are needed in the reconstruction (1.7) to approximate the solution. Consequently, the computational cost of the reduced-order model decreases.

The snapshots created for the minimum fluidization case were divided into two parts. The first part, which ranged from 0.2 to 0.35 seconds, included most of the transient part of the flow. The second part ranged from 0.35 to 1.0 seconds and included snapshots corresponding to the slower varying flow. Figure 43 shows the cumulative energy of the POD modes obtained using a single database that covered the entire time domain. Figures 44 and 45 show the cumulative energy of the POD modes for the split databases.

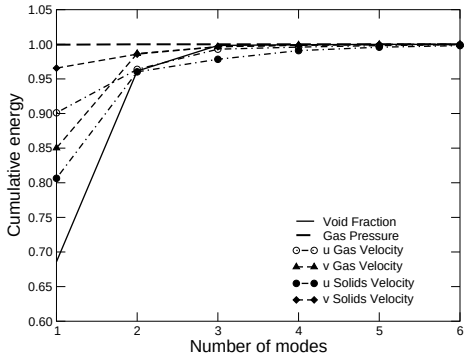


Fig. 43. Cumulative energy for a database that spans 0.2-1.0 seconds.

The energy variation extracted from the 0.2-1.0 seconds database, shown in Fig. 43, was similar to the energy variation extracted from the transient snapshots, shown in Fig. 44. Most of the energy extracted from the 0.35-1.0 seconds database was, however, concentrated in the first mode, as shown in Fig. 45. This concentration of the energy allowed capturing most of the flow features using fewer modes compared to the transient regime.

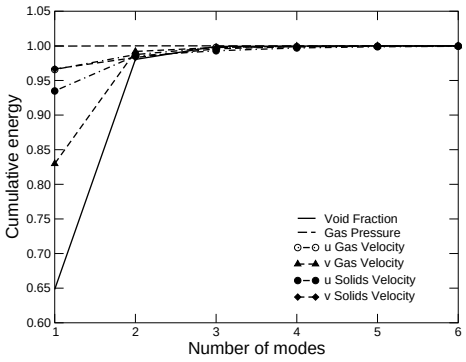


Fig. 44. Cumulative energy for a database that spans 0.2-0.35 seconds.

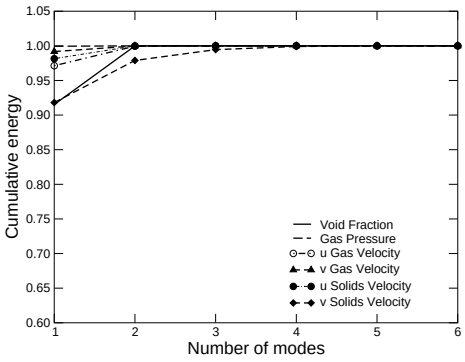


Fig. 45. Cumulative energy for a database that spans 0.35-1.0 seconds.

Computing the auto-correlation matrix for each database subset is a straight forward process. Determining the bounds of each subset such that to reduce the computational cost is less trivial. Two methods for the separation of the snapshots into subsets were used. The first method measured the time variation of the time coefficients, α, of the dominant modes of each field variable. The second method monitored the ratio between the variation of CPU time and the variation of physical time.

In the first method, the differences between the values of the coefficients were larger in the transient regime than the differences in the post transient period. In addition, in the post transient period, the variation of the time coefficients was almost constant. The advantage of this method is that the end of the transient regime can be detected accurately during calculation. The disadvantage of this method is that there is not a unique value that determines the limit of the transience for all six field variables. The difference values must be calculated and monitored for several modes for every field variables. Monitoring all of these values can, in some cases, produce conflicting information. An alternative to monitoring all six field variables

was to monitor only the field variables that most affect the flow. For the minimum fluidization case, the flow features were most affected by the first modes of gas pressure and gas velocity in the y-direction. A time history plot of the first time coefficient of gas pressure, $\alpha_1^{p_g}$, is shown in Fig. 46. The gas pressure time coefficients difference, $|\alpha_1^{p_g}(t+\Delta t)-\alpha_1^{p_g}(t)|$, is shown in Fig. 47. In this case, the end of the transient regime was at $t_{physical} = 0.35$ seconds.

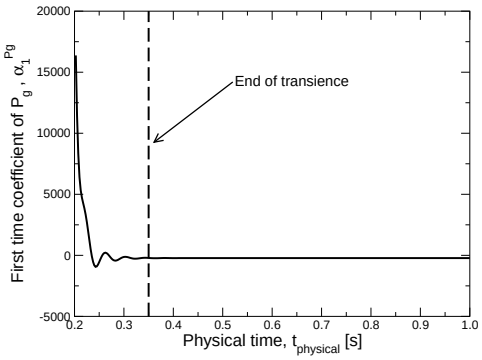


Fig. 46. Time history of the first time coefficient of gas pressure.

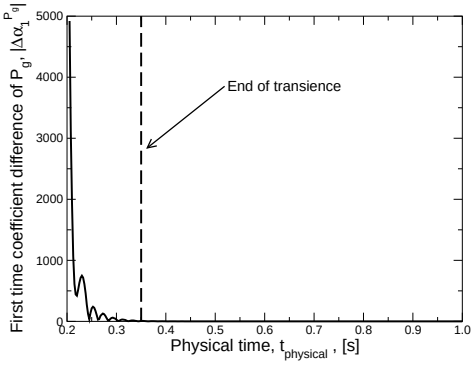


Fig. 47. Time history of the variation of the first time coefficient of gas pressure.

The second method proposed for separating the snapshots was to calculate the ratio of the change in CPU time and the change in physical time, $\Delta t_{CPU}/\Delta t_{physical}$. During transience, longer computation times are needed per time step as shown in Fig. 48. The advantage of monitoring this parameter is that the time slope is a single value that describes the behavior of all six field variables. The disadvantage of this method is that it over predicts the end of the transience. The magnitude of the slope decreased rapidly until $t_{physical} = 0.3$ seconds and continued to decrease somewhat slower to a quasi-constant value at $t_{physical} = 0.45$ seconds. Placing the end of the transient region at $t_{physical} = 0.45$ seconds is more conservative than the 0.35 seconds predicted by the first method.

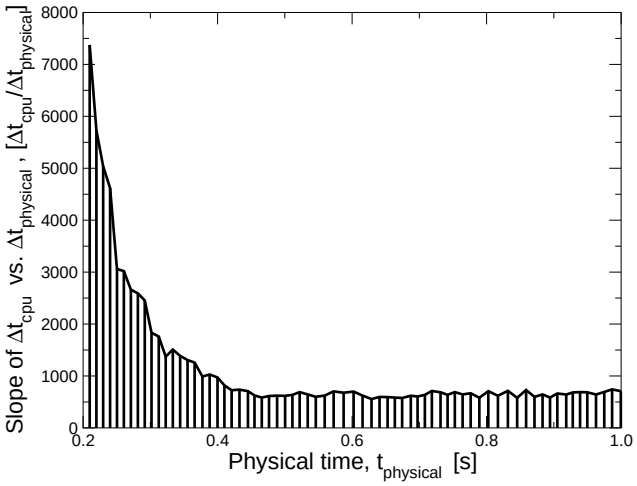


Fig. 48. Slope of the total CPU time vs physical time measured at 0.01 sec. increments.

Without database splitting, the best speed-up factor with qualitatively accurate results was 21 for the following number of modes: N_{p_g}= 2, $N_{ug} = 1$, $N_{vg} = 5$, $N_{us} = 4$, $N_{vs} = 3$ and $N_{\epsilon g} = 3$, for the minimum fluidization case. Splitting the database at $t_{physical} = 0.35$ seconds into two sets of basis functions resulted in a speed-up factor of 30. The increase in the speed-up factor was attributed to fewer modes being used for the post transient period.

2. Freezing the Matrix of the Linear System

The projection of the discretized differential equation onto the basis functions takes most of the computational time of a subiteration, as shown in Fig. 42(b). Numerical tests for a minimum fluidization case[9] showed that the components of the projected $\tilde{\mathcal{A}}$ matrix do not vary significantly past the transient period. To quantify the variation of the $\tilde{\mathcal{A}}$ matrix, the eigenvalues of the product, $\tilde{\mathcal{A}}^{-1}(t)\tilde{\mathcal{A}}(t + \Delta t)$, where Δt is the subiteration time step, were compared to the eigenvalues of the identity matrix. This

comparison was performed using $\tilde{\mathcal{A}}^{v_g}$ matrix extracted from the transient and post transient periods.

The eigenvalues of $\tilde{\mathcal{A}}^{v_g}$ varied by approximately $4.1 * 10^{-3}\%$ between time steps in the transient period. The eigenvalues varied by approximately $6.7 * 10^{7}\%$ between time steps in the period after transience. Table XVI shows the eigenvalues of $\tilde{\mathcal{A}}^{v_g}$ extracted from the transient and post transient periods. To reduce computational time, the $\tilde{\mathcal{A}}$ matrices of the linear systems (4.11)-(4.34) were no longer updated every subiteration. The $\tilde{\mathcal{B}}$ vectors, however, were updated every subiteration. Freezing of

Table XVI. Eigenvalues of $\tilde{\mathcal{A}}^{v_g}$ extracted from the transient and post transient periods.

$t_{physical}$[s]	λ_1	λ_2	λ_3	λ_4	λ_5
0.25	1.000000000	1.000000000	1.000000000	0.999982334	0.999936216
0.90	1.000000000	1.000000000	1.000000000	1.000000000	0.999999933

the matrix was implemented by calculating the $\tilde{\mathcal{A}}$ matrix for the first subiteration at the beginning of each time step. Then for the next subiteration the $\tilde{\mathcal{A}}$ matrix was held fixed and only the $\tilde{\mathcal{B}}$ vector in the $\tilde{\mathcal{A}}x = \tilde{\mathcal{B}}$ system (4.11)-(4.34) was recomputed. Freezing was only performed for every other subiteration. This application of freezing resulted in a 6.0% increase in error averaged over all the field variables and a 10.1% decrease of computational time. The increase in error and decrease of computational time were measured with respect to the case when freezing was not used. Errors of the ROM when freezing was not used, ϵ_{NF}, and with freezing, ϵ_F, compared to the full-order model solution are shown in Table XVII. The error, ϵ, for gas pressure, p_g, was defined as:

$$\epsilon^{p_g} = \frac{\sum_N (p_g^{ODEx} - p_g^{MFIX})}{N * 101000} \tag{6.1}$$

where N is the total number of points in the spatial domain. The error is normalized by the pressure at the outlet 101000 Pa. Similarly, the velocity errors and solids volume fraction errors are normalized by the gas inlet velocity 12.6 cm/s and the maximum value of ϵ_s, 1.0, respectively. The percentage difference of error between the two ROM, ϵ^D, cases is defined as:

$$\epsilon^D = 100 * \frac{|\epsilon_{NF} - \epsilon_F|}{\epsilon_{NF}} \tag{6.2}$$

Table XVII. ROM errors at $t_{physical} = 1.0$ seconds with and without freezing.

Variable	$\epsilon_{NF}[\%]$	$\epsilon_F[\%]$	ϵ^D [%]
u_g	$1.655 * 10^{-3}$	$1.649 * 10^{-3}$	0.362
v_g	0.068	0.072	5.882
u_s	$3.361 * 10^{-4}$	$3.322 * 10^{-4}$	1.160
v_s	$1.452 * 10^{-4}$	$1.782 * 10^{-4}$	22.727
p_g	$1.555 * 10^{-3}$	$1.644 * 10^{-3}$	5.723
ϵ_s	$3.551 * 10^{-5}$	$3.541 * 10^{-5}$	0.282
Avg.	0.012	0.013	6.023

The relationship between the computational time and the physical time was used herein to determine when to freeze the matrix. Figure 49 shows a plot of this relationship for the minimum fluidization case. After $t_{physical} = 0.4$ seconds the relationship between the iteration time and physical time was close to linear. This indicated that the transient period had ended.

Applying the freezing algorithm to ODEx at the physical time of 0.4 seconds resulted in a 10% decrease in computational time. Given that freezing the projection

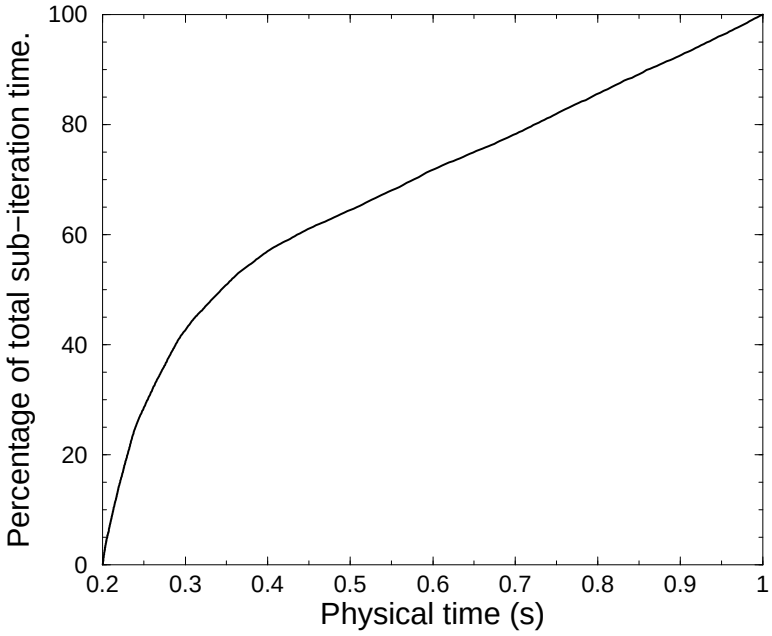


Fig. 49. Relationship between physical time and CPU time.

of the $\tilde{\mathcal{A}}$ matrix greatly reduces the number of operations per subiteration, a 10% decrease in required CPU time is a small improvement. This reduced gain was due to an increase in the number of subiterations, which diminished the benefit of saving computational time by not updating the $\tilde{\mathcal{A}}$ matrix.

3. Time Step Adjustment

MFIX and ODEx use an identical method to adjust the time step of the time integration. During integration time step size adjustment is determined by the speed of convergence of the calculation. Both codes use two parameters to adjust the time step during calculation: (1) the frequency of the time step adjustment and (2) the size of the time step adjustment. Given identical initial time steps and time step adjustment parameters, the time step size in ODEx increased by at least one order of magnitude, while the time step in MFIX remained almost constant, as shown in Fig. 50.

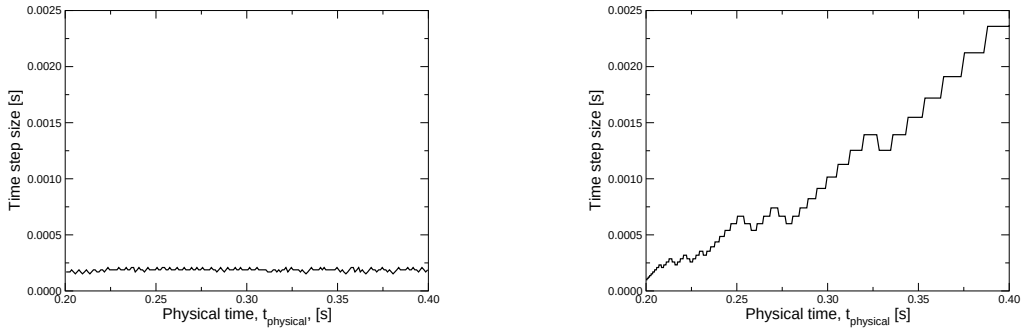


Fig. 50. Time step size vs. physical time: a) MFIX, and b) ODEx.

The time step cannot grow unbounded because the time accuracy of the results would diminish. The maximum time step size can be manually set based on the time scale of the simulated flow phenomena. Three time step parameters were modified in ODEx to achieve a faster computation time: (1) the frequency of the time step adjustment, (2) the size of the time step adjustment and (3) the size of the initial time step.

It was found through numerical tests that the most effective time step adjustment to speed up ODEx was to increase the initial time step size. Time step size growth during the transient region was more limited than in the post transient period to satisfy convergence. As shown in Fig. 50b the time step did not grow steadily in the initial part of the calculation. Due to this restriction in the transient period, increasing the initial time step resulted in a limited computational speed up. For example, changing the initial time step size from 10^{-4} to 10^{-3} seconds reduced the computational time by approximately 9% for any mode combination for the minimum fluidization case. This relatively small reduction in computational time did not reflect the order of magnitude decrease of the initial time step. Much of the computational speed up was lost because it was necessary to reduce the larger initial time step size to satisfy convergence during the initial transient calculations.

4. Summary of Acceleration Methods

A summary of the speed-up factors achieved by the various versions of the reduced-order model ODEx is given in Table XVIII. The speed-up factors are reported with respect to the full-order model code, MFIX. All the ODEx versions used the set of modes: N_{p_g}= 2, $N_{ug} = 1$, $N_{vg} = 5$, $N_{us} = 4$, $N_{vs} = 3$ and $N_{\epsilon g} = 3$.

Table XVIII. Speed-up factors: ODEx with acceleration methods vs. MFIX.

	Speed-up Factor
MFIX	1
ODEx with no acceleration method	21
ODEx with database splitting	30
ODEx with projection freezing	23
ODEx with time step adjustment	25
ODEx with database splitting and time step adjustment	114

Restriction of time step size growth caused by the initial transient calculations led to the combination of database splitting and adjustment of the initial time step size. First the database was split into a transient and post transient period. This allowed the use of a small initial time step for the transient calculations and a relatively large initial time step and fewer modes for the post transient calculations. The combination of these two methods resulted in a two orders of magnitude speed-up factor of ODEx compared to MFIX for the minimum fluidization case.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

This chapter presents the conclusions derived from this work. In addition, recommendations for future work are presented.

A. Conclusions

In this thesis a POD-based reduced-order model has been developed to simulate isothermal and non-isothermal transport phenomena in a fluidized bed. Results were presented for both the POD-based ROM and full-order model for comparison. The presented cases used for comparison were a minimum fluidization case, a time extrapolation case using the POD-based ROM and an unsteady, periodic inlet boundary condition case. The results show that the POD-based ROM produces qualitatively accurate results while only requiring a few basis functions for each field variable.

Additionally this thesis presented four acceleration methods for reduced-order models based on the proper orthogonal decomposition method: (i) database splitting, (ii) an algorithm for solving quasi-symmetrical matrices, (iii) freezing the matrix of linear system, and (iv) a time step adjustment.

The database splitting algorithm used the fact that fewer modes were needed for reconstruction when the integration was divided. The snapshots created for the minimum fluidization case were divided into two parts. The first part, which ranged from 0.2 to 0.35 seconds, included most of the transient part of the flow. The second part ranged from 0.35 to 1.0 seconds and included snapshots corresponding to the slower varying flow. Compared to the first integration period, relatively few modes were needed for the second period of integration.

The matrix freezing approach increased the computational efficiency by reducing

the number of operations performed during the post transient calculations. Although the $\tilde{\mathcal{A}}$ matrix was not for every subiteration, the right hand side vector, $\tilde{\mathcal{B}}$, was updated every subiteration because it was necessary and computationally inexpensive. Only a small decrease in the computation time (10%) was obtained, since the number of subiterations increased in order to achieve a converged solution.

Time step adjustments used the fact that the time step size could be increased to a larger value in the POD-based ROM than in MFIX. It was found through numerical tests that the most effective method to speed up the ROM was to adjust the initial time step size. To overcome relatively strict time step size limitations during the transient period, the time step adjustment and database splitting acceleration techniques were used together. This combination of methods achieved a two orders of magnitude increase in computational efficiency.

B. Future Work

The challenge for the future is to apply the POD-based ROM developed in this thesis to more dynamic cases with larger amounts of flow field unsteadiness. Additionally POD may be applied to the field variables as a whole instead of individually. The lessons learned from this investigation may eventually be applied to generate a POD-based reduced-order model of aeroelastic phenomena.

REFERENCES

[1]Hayashi, C., *Nonlinear Oscillations in Physical Systems*, McGraw-Hill, New York, 1964, pp. 28–30.

[2]Lucia, D.J., Beran, P.S., and Silva, W. A., "Reduced-Order Modeling: New Approaches for Computational Physics," *Progress in Aerospace Sciences*, Vol. 40, 2004, pp. 51–117.

[3]Rugh, W.J., *Nonlinear System Theory: The Volterra/Wiener Approach*, The Johns Hopkins University Press, Baltimore, 1981, pp. 3–21.

[4]Beran, P. S., "Computation of Limit Cycle Oscillation Using a Direct Method," AIAA 98–1462, April 1999.

[5]Dowell, E.H., Thomas, J.P., and Hall, K.C., "Computation of Unsteady Nonlinear Flows in Cascades Using a Harmonic Balance Technique," *Ninth International Symposium on Unsteady Aerodynamics, Aeroacoustics and Aeroelasticity of Turbomachines and Propellers*, Lyon, France, September 4–8, 2000.

[6]Silva, W.A., and Raveh, D.E., "Development of Aerodynamic/Aeroelastic State-space from CFD-based Pulse Responses," AIAA 2001-1213, 42nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, Seattle WA, April 16–19, 2001.

[7]Lucia D.J., and Beran P.S., "Reduced-Order Model Development Using Proper Orthogonal Decomposition and Volterra Theory," AIAA 0001-1452, vol. 42, no. 6, 2004, pp. 1181–1190.

[8]Boyd S.P., "Volterra Series: Engineering Fundamentals," Ph.D. dissertation, University of California, Berkely, 1985.

[9]Yuan, T. "Reduced Order Modeling for Transport Phenomena based on Proper

Orthogonal Decomposition," Master's Thesis, Texas A&M University, Dec. 2003, pp. 4–24.

[10]Berkooz, G., Holmes, P., and Lumley, J. L., "The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows," *Annual Review of Fluid Mechanics*, Vol. 25, 1993, pp. 539–575.

[11]Sirovich, L., "Turbulence and the Dynamics of Coherent Structures Part:I-III," *Quarterly of Applied Mathematics*, Vol. XLV, No. 3, 1987, pp. 561–590.

[12]Chambers, D. H., Adrian, R. J., Moin, P., Stewart, D. S., and Sung, H. J., "Karhunen-Loéve Expansion of Burgers' Model of Turbulence," *Physics of Fluids*, Vol. 31, No. 9, 1988, pp. 2573–2582.

[13]Sahan, R. A., Liakopoulos, A., and Gunes, H., "Reduced Dynamical Models of Nonisothermal Transitional Grooved-Channel Flow," *Physics of Fluids*, Vol. 9, No. 3, 1997, pp. 551–565.

[14]Cizmas, P. G. A., and Palacios, A., "Proper Orthogonal Decomposition of Turbine Rotor-Stator Interaction," *Journal of Propulsion and Power*, Vol. 19, No. 2, 2003, pp. 268–281.

[15]Liu, Z.-C., Adrian, R. J., and Hanratty, T. J., "Reynolds Number Similarity of Orthogonal Decomposition of the Outer Layer of Turbulent Wall Flow," *Physics of Fluids*, Vol. 6, No. 8, 1994, pp. 2815–2819.

[16]Cizmas, P. G., Palacios, A., O'Brien, T., and Syamlal, M., "Proper-Orthogonal Decomposition of Spatio-Temporal Patterns in Fluidized Beds," *Chemical Engineering Science*, Vol. 58, No. 19, 2003, pp. 4417–4427.

[17]Syamlal, M., Rogers, W., and O'Brien, T., "MFIX Documentation: Theory Guide," Technical Note DOE/METC-94/1004, U.S. Department of Energy, Morgantown, West Virginia, 1993.

[18]Syamlal, M., "MFIX Documentation: Numerical Technique," EG&G Technical Report DE-AC21-95MC31346, U.S. Department of Energy, Morgantown, West Virginia, 1998.

[19]Holmes, P. J., Lumley, J. L., Berkooz, G., Mattingly, J. C., and Wittenberg, R. W., "Low-Dimensional Models of Coherent Structures in Turbulence," *Physics Reports*, Vol. 287, 1997, pp. 337–384.

[20]Yuan, T., Cizmas, P.G., O'Brien, T., "A reduced-order model for a bubbling fluidized bed based on proper orthogonal decomposition," *Computers and Chemical Engineering*, Vol. 30, 2005 pp 243–259.

[21]Cizmas, P. G. A. and Palacios, A., "Proper Orthogonal Decomposition of Turbine Rotor-Stator Interaction," AIAA Journal of Propulsion and Power, *AIAA Journal*, Vol. 19, No. 2, 2003, pp. 268–281.

[22]Press, W. H., Vetterling, W. T., Teukolsky, S. A., and Flannery, B. P., "Numerical Recipes in FORTRAN - The Art of Scientific Computing," Cambridge, 1992.

[23]Cizmas, P. G. A., "An Acceleration Approach for Reduced-Order Models Based on Proper Orthogonal Decomposition," 45th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January, 2007.

[24]Oliveira, K. and Anastasiou, An Efficient Computational Model for Water Wave Propogation in Costal Regions, *Applied Ocean Res.*, Vol. 20, No. 5, 1998, pp. 263–271.

APPENDIX A

CONSTITUTIVE MODELS

Gas phase stress tensor

The gas viscous stress tensor $\bar{\bar{\tau}}_g$ is assumed to be of the Newtonian form

$$\bar{\bar{\tau}}_g = 2\mu_g \bar{\bar{D}}_g - \lambda_g \mathrm{tr}(\bar{\bar{D}}_g)\bar{\bar{I}},$$

where μ_g is the gas phase viscosity; $\lambda_g = -2/3\mu_g$; $\bar{\bar{I}}$ is an identity tensor; $\bar{\bar{D}}_g$ is the gas phase strain rate tensor, given by

$$\bar{\bar{D}}_g = \frac{1}{2}\left[\bigtriangledown \vec{v}_g + (\bigtriangledown \vec{v}_g)^T\right].$$

Solid phase stress tensor

MFIX uses the following model to compute the solid phase stress tensor

$$\bar{\bar{\tau}}_s = \begin{cases} \bar{\bar{\tau}}_s^{\mathcal{P}} & \text{if } \epsilon_g \leq \epsilon_g^*\text{: Plastic Regime} \\ \bar{\bar{\tau}}_s^{\mathcal{V}} & \text{if } \epsilon_g > \epsilon_g^*\text{: Viscous Regime} \end{cases},$$

where ϵ_g^* is the packed-bed void fraction at which a granular flow regime transition is assumed to occur and ϵ_g^* is usually set to the void fraction at minimum fluidization.[17] The superscript $\mathcal{P}$ stands for plastic regime and $\mathcal{V}$ for viscous regime.

- *Plastic Regime*:

$$p_s^{\mathcal{P}} = 10^{25}(\epsilon_g^* - \epsilon_g)^{10}$$

$$\bar{\bar{\tau}}_s^{\mathcal{P}} = 2\mu_s^{\mathcal{P}} \bar{\bar{D}}_s$$

$$\mu_s^{\mathcal{P}} = \frac{p_s^{\mathcal{P}} \sin\phi}{2\sqrt{I_{2D_s}}}$$

Herein $\bar{\bar{D}}_s$ denotes the solid phase strain rate tensor. ϕ is the angle of internal friction. I_{2D_s} is the second invariant of the deviator of $\bar{\bar{D}}_s$:

$$I_{2D_s} = \frac{1}{6}\left[(D_{s11} - D_{s22})^2 + (D_{s22} - D_{s33})^2 + (D_{s33} - D_{s11})^2\right] + D_{s12}^2 + D_{s23}^2 + D_{s31}^2.$$

- *Viscous Regime*:

$$p_s^{\mathcal{V}} = K_{1s}\epsilon_s^2\Theta_s$$

$$\bar{\bar{\tau}}_s^{\mathcal{V}} = 2\mu_s^{\mathcal{V}}\bar{\bar{D}}_s + \lambda_s^{\mathcal{V}}\mathrm{tr}(\bar{\bar{D}}_s)\bar{\bar{I}}$$

$$\lambda_s^{\mathcal{V}} = K_{2s}\epsilon_s\sqrt{\Theta_s}$$

$$\mu_s^{\mathcal{V}} = K_{3s}\epsilon_s\sqrt{\Theta_s}$$

$$K_{1s} = 2(1+e_s)\rho_s g_{0_s}$$

$$K_{2s} = 4d_{ps}\rho_s(1+e_s)\epsilon_s g_{0_s}/(3\sqrt{\pi}) - \frac{2}{3}K_{3s}$$

$$K_{3s} = \frac{d_{ps}\rho_s\sqrt{\pi}}{6(3-e_s)}\left[1 + 0.4(1+e_s)(3e_s-1)e_s g_{0_s}\right] + \frac{d_{ps}\rho_s 8\epsilon_s g_{0_s}(1+e_s)}{10\sqrt{\pi}}$$

$$K_{4s} = \frac{12(1-e_s^2)\rho_s g_{0_s}}{d_{ps}\sqrt{\pi}}$$

$$g_{0_s} = \frac{1}{1-\epsilon_s} + 1.5\epsilon_s\left(\frac{1}{1-\epsilon_s}\right)^2 + 0.5\epsilon_s^2\left(\frac{1}{1-\epsilon_s}\right)^3$$

Herein e_s is the coefficient of restitution for particle-particle collisions. d_{ps} is the solid particle diameter. The granular temperature Θ_s is given by

$$\Theta_s = \left\{\frac{-K_{1s}\epsilon_s\mathrm{tr}(\bar{\bar{D}}_s) + \sqrt{K_{1s}^2\mathrm{tr}^2(\bar{\bar{D}}_s) + 4K_{4s}\epsilon_s\left[K_{2s}\mathrm{tr}^2(\bar{\bar{D}}_s) + 2K_{3s}\mathrm{tr}(\bar{\bar{D}}_s^2)\right]}}{2\epsilon_s K_{4s}}\right\}^2$$

Gas-solid momentum transfer

$$F_{gs} = \frac{3\rho_g \epsilon_s \epsilon_g}{4 V_{rs}^2 d_{ps}} \left(0.63 + 4.8\sqrt{V_{rs}/Re_s}\right)^2 |\vec{v}_s - \vec{v}_g|$$

$$V_{rs} = 0.5\left(A - 0.06Re_s + \sqrt{(0.06Re_s)^2 + 0.12Re_s(2B - A) + A^2}\right)$$

$$A = \epsilon_g^{4.14}$$

$$B = \begin{cases} 0.8\epsilon_g^{1.28} & \text{if } \epsilon_g \leq 0.85 \\ \epsilon_g^{2.65} & \text{if } \epsilon_g > 0.85 \end{cases}$$

$$Re_s = \frac{d_{ps} |\vec{v}_s - \vec{v}_g| \rho_g}{\mu_g}$$

Gas-solids heat transfer

The gas-solids heat transfer coefficient γ_{gm} quantifies the heat transfer between the gas and solids phases. The term γ_{gm}^0 is the gas-solids heat transfer coefficient not corrected for the interphase mass transfer. MFIX uses the following equation to calculate the gas-solids heat transfer coefficient

$$\gamma_{gm} = \frac{C_{pg} R_{0m}}{exp\left(\frac{C_{pg} R_{0m}}{\gamma_{gm}^0}\right) - 1}$$

$$\gamma_{gm}^0 = \frac{6 k_g \epsilon_{sm} Nu_m}{d_{pm}^2}$$

$$Nu_m = (7 - 10\epsilon_g + 5\epsilon_g^2)(1 + 0.7 Re_m^{0.2} Pr^{1/3}) + (1.33 - 2.4\epsilon_g + 1.2\epsilon_g^2) Re_m^{0.7} Pr^{1/3}$$

where the subscript m denotes the gas or solids phase, Nu_m is the Nusselt number, Pr is the Prandtl number and R_{0m} is the rate of transfer of mass from the m^{th} solids phase to the 0^{th} (or gas) phase.

Gas and solids conduction

MFIX uses the following equations to model the gas and solids phase conduction:

$$\vec{q}_g = -k_g \bigtriangledown T_g$$

$$\vec{q}_{sm} = -k_{sm} \bigtriangledown T_{sm}$$

APPENDIX B

SAMPLE INPUT FILE FOR ODEX

```
# ODEx input file for Case I
#
# 1.1 Run control section
#
  TSTART  = 0.2028
  TSTOP = 1.0
  DT    = 1.D-4
  MAX_NIT = 30
  DT_MAX  = 1.D0
  DT_MIN  = 1.D-6
  DT_FAC  = 0.9D0
  TOL_RESID = 1.D-4
  TOL_DIVERGE = 1.D2
#
# 1.2 Geometry and discretization section
#
  XLENGTH = 25.4D0     IMAX = 108
  YLENGTH = 76.5D0     JMAX = 124
  DISCRETIZE = 2
#
# 1.3 Physical properties section
#
  MU_g0   = 1.8D-4
  MW_g0   = 29.D0
  T_g0    = 297.D0
  RO_s0   = 1.00
  D_p     = 0.05
  C_e     = 0.8
  Phi     = 30.0
  EP_star = 0.4
#
# 1.4 POD mode section
#
  nP_g    = 2
  nU_g    = 2
  nV_g    = 5
  nU_s    = 8
  nV_s    = 6
  nEP_g   = 7
```

APPENDIX C

SAMPLE INPUT FILE FOR ODET

```
#
# ODEt input file for Case II
#
# 1.1 Run control section
#
  TSTART  = 0.2028
  TSTOP = 1.0
  DT    = 1.D-4
  MAX_NIT = 30
  DT_MAX  = 1.D0
  DT_MIN  = 1.D-6
  DT_FAC  = 0.9D0
  TOL_RESID = 1.D-4
  TOL_DIVERGE = 1.D2
#
# 1.2 Geometry and discretization section
#
  XLENGTH = 25.4D0     IMAX = 108
  YLENGTH = 76.5D0     JMAX = 124
  DISCRETIZE = 2
#
# 1.3 Physical properties section
#
  MU_g0   = 1.8D-4
  MW_g0   = 29.D0
  T_g0    = 297.D0
  RO_s0   = 1.00
  C_PG0   = 0.25
  GAMA_RG0 = 0.0
  GAMA_RS0 = 0.0
  D_p     = 0.05
  C_e     = 0.8
  Phi     = 30.0
  EP_star = 0.4
#
# 1.4 POD mode section
#
  nP_g    = 2
```

```
nU_g    = 2
nV_g    = 5
nU_s    = 8
nV_s    = 6
nEP_g   = 7
nT_g    = 9
nT_s    = 3
```

VITA

Brian R. Richardson, was born in Waco, Texas, USA. He received his Bachelor of Science degree in Aerospace Engineering from the Texas A&M University, College Station, Texas in May 2005.

He began his graduate study at Texas A&M University in June 2005 and received his Masters of Science degree in Aerospace Engineering in May 2008. His research interests focused on the reduced-order modeling of transport phenomena and heat transfer for multi-phase flows.

Brian R. Richardson's permanent address is 4832 Scottwood Drive, Waco, Texas 76708, USA.

The typist for this thesis was Brian R. Richardson.